INSTITUTE FOR ADVANCED COMPUTING

AND SOFTWARE DEVELOPMENT

AKURDI, PUNE – 411044

Documentation On

"Sarcasm Detection using NLP"

PG-DBDA FEB 2025

Submitted By:

Group No: 1

Prajakta Kumbhalkar (252525)
Pratiksha Gunjal (252534)
Sunidhi Choudhary (252548)

Dr. Shantanu Pathak — **Project Guide**

Mr. Prashant Deshpande — **Centre Coordinator**

DECLARATION

I, the undersigned hereby declare that the project report titled " Sarcasm Detection using NLP" written and submitted by me to Institute For Advanced Computing And Software Development Akurdi Pune, in the fulfillment of requirement for the award of degree of Post Graduate Diploma In Big Data Analytics (PG DBDA) under the guidance of Dr. Shantanu Pathak Sir. It is my original work I have not copied any code or content from any source without proper attribution, and I have not allowed anyone else to copy my work. The project was completed using Python and ML and libraries and NLP. The project was developed as part of my academic coursework. I also confirm that the project is original, and it has not been submitted previously for any other academic or professional purpose.

Place:	Signature:
Date:	Name: Prajkata Kumbhalkar /Pratiksha Gunjal/ Sunidhi Choudhary

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Dr. Shantanu Pathak Sir, Project Guide , for providing me with the guidance and support to complete this academic project . His valuable insights , expertise and encouragement have been instrumental in the success of this project.

I would also like to thank my fellow classmates for their support and cooperation during the project .Their feedback and suggestions were helpful in improving the quality of the project.

I would like to extend my gratitude to Mr. Prashant Deshpande Sir, Centre Coordinator , for providing me with the necessary resources and facilities to complete this project . His support has been crucial in the timely completion of this project.

Finally , I would like to thank my family and friends for their constant encouragement and support throughout the project . Their belief in me has been a constant source of motivation and inspiration.

Thank you all for your support and guidance in completing this academic project.

ABSTRACT

Sarcasm Detection using Natural Language Processing (NLP) aims to accurately identify sarcastic statements in text, a task that is often challenging due to subtle linguistic cues and context dependency. This project focuses on building a robust classification model to detect sarcasm in news headlines.

The dataset, composed of labelled headlines, was analysed for text patterns and linguistic structure. Preprocessing steps included tokenization, padding, and attention to sequence length. Two models were evaluated: a traditional LSTM (Long Short-Term Memory) network and BERT (Bidirectional Encoder Representations from Transformers), a transformer-based language model.

BERT outperformed LSTM in both accuracy and contextual understanding, owing to its deep attention mechanism and pre-trained knowledge of language. The fine-tuned BERT model achieved high classification accuracy on the validation set, demonstrating strong performance in recognizing sarcastic tone and irony.

To enhance deployment readiness, the trained model and tokenizer were saved and exported. This project highlights the effectiveness of transformer-based models in sentiment-related tasks and showcases how AI can be used to interpret nuanced human communication in text.

Table Of Contents

1. **INTRODUCTION**...1

1.1 Problem Statement... 1

1.2 Scope...2

1.3 Aim & Objectives...3

2. **PROJECT DESCRIPTION**..4

2.1 Project work flow 4

2.2 Data Collection... 5

2.3 Studying the Data...5

2.4 Studying the Model...5

2.5 Implementing the Model.. 6

2.6 Validating the Model... 7

3. **MODEL DESCRIPTION**..10

3.1 ML model..10

3.2 Why LSTM & BERT.. 11

4. **DATA FLOW**..12

4.1 Data flow of project...12

5. **PROJECT REQUIREMENTS** 13

6. **FUTURE SCOPE**...15

7. **CONCLUSION**.. 16

8. **REFERENCES**..17

Table of Figure:

Figure 1 ***Project work flow diagram***......4
Figure 2 ***Features Descriptions***......5
Figure 3 ***Code for Model Implementation***......7
Figure 4 ***Validation***......8
Figure 5 ***Output & LSTM & BERT report***...... 16

CHAPTER 1
INTRODUCTION

In the era of social media, online reviews, and digital communication, understanding the true intent behind a text has become a significant challenge for both individuals and automated systems. Among the various forms of expression, **sarcasm** poses a unique problem—it often conveys the opposite of the literal meaning, making it difficult for traditional sentiment analysis models to interpret+ correctly. For example, the sentence *"Oh, great! Another Monday morning…"* may appear positive due to words like "great," but in context, it is likely negative.

Detecting sarcasm is important in areas such as **customer feedback analysis**, **social media monitoring**, **opinion mining**, and **chatbots**, where accurate sentiment detection can influence decision-making, brand perception, and user experience.

This project focuses on **Natural Language Processing (NLP)** to identify sarcasm in text using **two distinct machine learning approaches**:

1. **Long Short-Term Memory (LSTM)** – A deep learning model that excels at capturing sequential dependencies in text.
2. **Bi-directional Encoder Representations from Transformers (BERT)** – A transformer-based model capable of understanding contextual meaning in both directions of a sentence.

By comparing these models, the project aims to explore the effectiveness of sequential vs. transformer-based architectures in sarcasm detection.

1. Problem Statement

Sarcasm detection is a challenging subtask of sentiment analysis due to the **mismatch between the literal and intended meaning** of text. Most traditional sentiment analysis models rely heavily on surface-level word polarity, which often fails when sarcasm is present. This misinterpretation can lead to incorrect sentiment classification, thereby reducing the effectiveness of automated text analysis systems.

The core problem addressed in this project is:

How can NLP models effectively detect sarcasm in text to improve the accuracy of sentiment analysis systems?

Key challenges include:

Ambiguity of sarcasm – The same sentence can be sarcastic or genuine depending on context.

Context dependency – Sarcasm often requires prior knowledge or conversational context to be understood.

Data imbalance – Sarcastic sentences are less frequent compared to non-sarcastic ones in natural datasets.

This project proposes using **LSTM** and **BERT** models to learn linguistic and contextual cues that differentiate sarcastic text from literal statements, with the goal of improving sarcasm classification performance.

2. Scope

This project is limited to detecting sarcasm in **short textual statements**, such as social media posts, news headlines, and comments, using machine learning techniques. It focuses on two deep learning models—**LSTM** and **BERT**—to analyze, classify, and compare their performance in sarcasm detection.
The scope includes:

Pre-processing text data (tokenization, cleaning, embedding).

Training and evaluating two separate models—LSTM and BERT.

Comparing the models based on performance metrics like accuracy, precision, recall, and F1-score.

Providing insights on which approach is more suitable for sarcasm detection in short text datasets.

This study does not cover:

Audio or video-based sarcasm detection.

Multilingual sarcasm detection (focus is on English text).

Contextual sarcasm detection from multi-turn conversations beyond the given text.

3.Aim & Objective

The primary aim of this project is:

- **To design and compare deep learning-based NLP models for detecting sarcasm in textual data, thereby improving sentiment analysis accuracy.**
- To achieve the project aim, the following objectives are outlined:
- To achieve the aim, the following objectives are set:

1. **Understand the nature of sarcastic text** and the challenges it poses for NLP-based sentiment analysis.
2. **Preprocess and prepare the dataset** for sarcasm detection, ensuring quality and consistency.
3. **Implement an LSTM-based model** to learn sequential patterns and dependencies in text.
4. **Implement a BERT-based model** to capture contextual and bidirectional relationships in text.
5. **Train and evaluate both models** using standard performance metrics (Accuracy, Precision, Recall, F1-score).
6. **Compare the results** to determine which model performs better for sarcasm detection.
7. **Draw conclusions and suggest future improvements** for sarcasm detection systems.

CHAPTER 2
PROJECT DESCRIPTION

1.Project Workflow

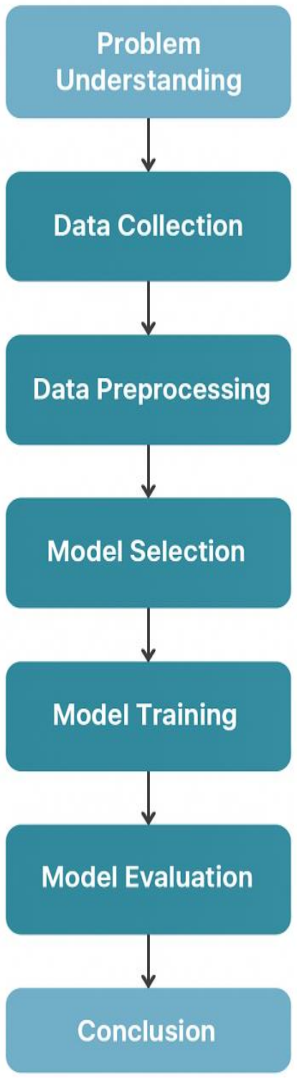


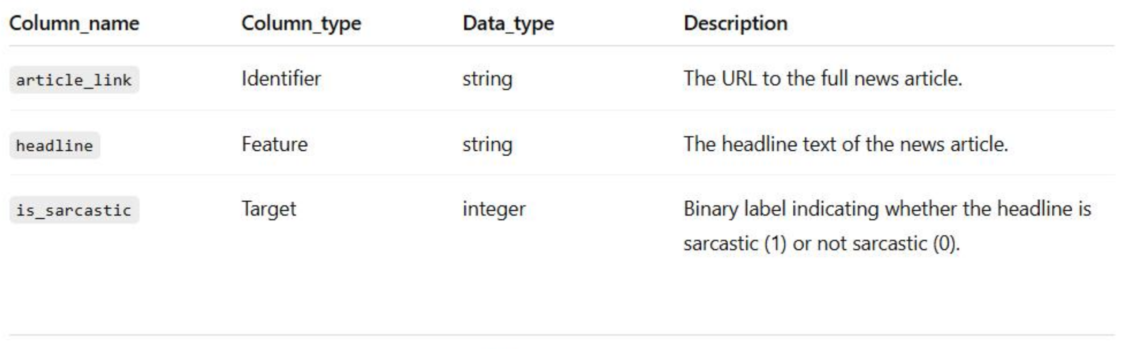
Fig 1: Project work-flow diagram

2.Data Collection

The dataset used in this project is sourced from Hugging Face Datasets, consisting of 55,328 labeled headlines. Each record contains:
Headline Text – The actual news headline or statement.
Label – 1 for sarcastic, 0 for non-sarcastic.
The dataset provides a balanced and diverse collection of sarcastic and non-sarcastic samples from multiple news sources and online platforms, making it suitable for training and evaluating sarcasm detection models.

3.Studying the data

Before modeling, the dataset is explored to understand its structure and characteristics:
Class Distribution – Ensure balanced representation of sarcastic and non-sarcastic labels.
Text Length Analysis – Check average word counts to guide tokenization and sequence length settings.
Common Words & Phrases – Identify frequent words in sarcastic vs. non-sarcastic text using word clouds.
Sentiment Patterns – Observe polarity distribution to see how sarcasm skews sentiment analysis results.
This study reveals the need for context-aware models, as sarcastic text often uses positive words in a negative sense.



Column_name	Column_type	Data_type	Description
`article_link`	Identifier	string	The URL to the full news article.
`headline`	Feature	string	The headline text of the news article.
`is_sarcastic`	Target	integer	Binary label indicating whether the headline is sarcastic (1) or not sarcastic (0).

Fig 2: *Features Descriptions*

4.Studying the Model

Two models are studied and selected for implementation:
LSTM (Long Short-Term Memory) – A recurrent neural network architecture that captures long-term dependencies and sequence patterns in text.

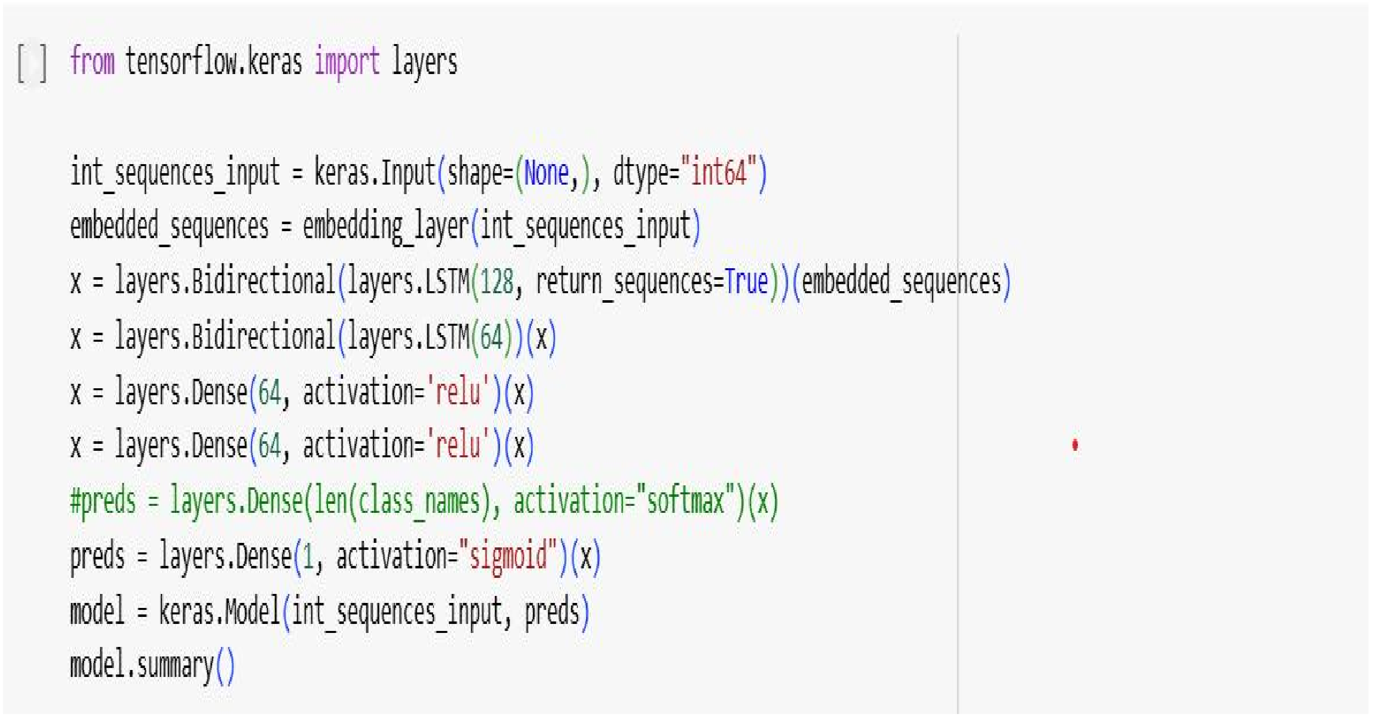
Strength: Good at learning sequential word dependencies
Limitation: Less effective in capturing bidirectional context.

BERT (Bidirectional Encoder Representations from Transformers) – A transformer-based model pre-trained on large corpora, capable of understanding context from both left and right sides of a word.

Strength: Captures deep contextual meaning and word relationships.
Limitation: Computationally expensive compared to LSTM.

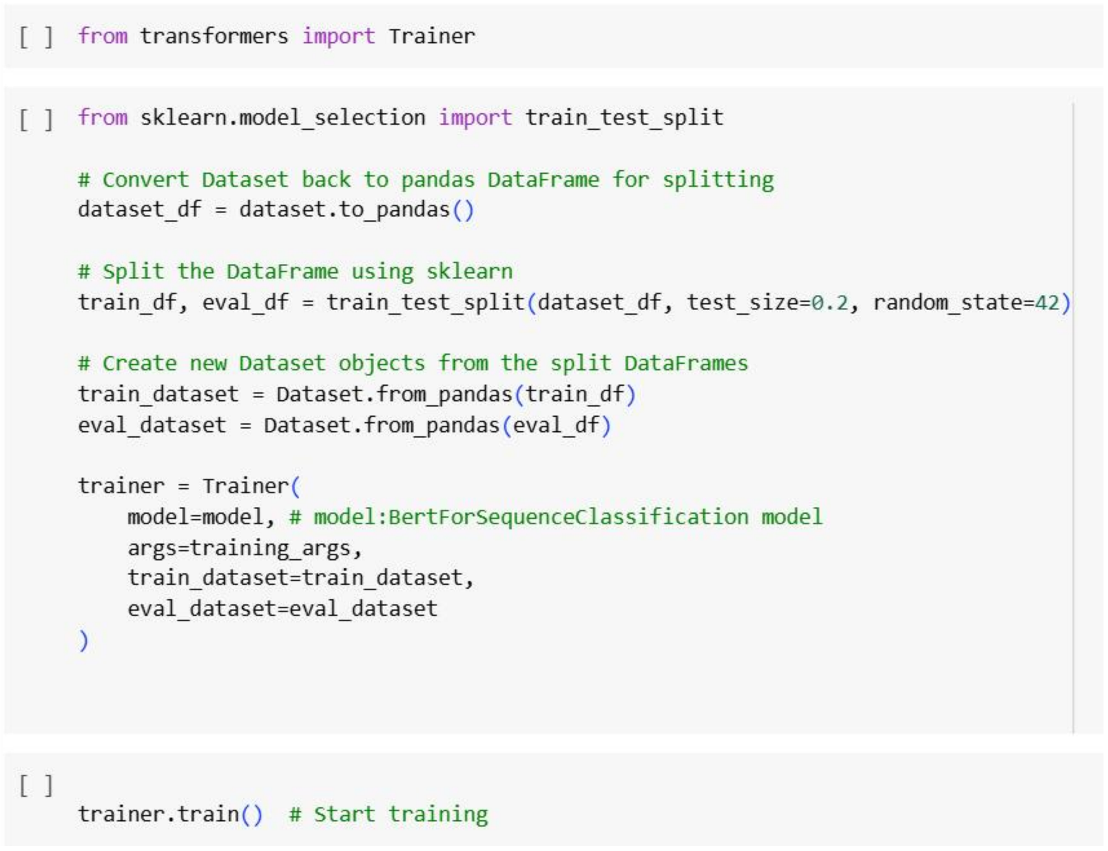
5 Implementing the Model

LSTM



```
[ ] from tensorflow.keras import layers

    int_sequences_input = keras.Input(shape=(None,), dtype="int64")
    embedded_sequences = embedding_layer(int_sequences_input)
    x = layers.Bidirectional(layers.LSTM(128, return_sequences=True))(embedded_sequences)
    x = layers.Bidirectional(layers.LSTM(64))(x)
    x = layers.Dense(64, activation='relu')(x)
    x = layers.Dense(64, activation='relu')(x)
    #preds = layers.Dense(len(class_names), activation="softmax")(x)
    preds = layers.Dense(1, activation="sigmoid")(x)
    model = keras.Model(int_sequences_input, preds)
    model.summary()
```

BERT



```
[ ] from transformers import Trainer
```

```
[ ] from sklearn.model_selection import train_test_split

    # Convert Dataset back to pandas DataFrame for splitting
    dataset_df = dataset.to_pandas()

    # Split the DataFrame using sklearn
    train_df, eval_df = train_test_split(dataset_df, test_size=0.2, random_state=42)

    # Create new Dataset objects from the split DataFrames
    train_dataset = Dataset.from_pandas(train_df)
    eval_dataset = Dataset.from_pandas(eval_df)

    trainer = Trainer(
        model=model, # model:BertForSequenceClassification model
        args=training_args,
        train_dataset=train_dataset,
        eval_dataset=eval_dataset
    )
```

```
[ ]
    trainer.train()  # Start training
```

Fig 3: Code for Model Implementation

2.6 Validating the Model

LSTM Implementation Steps:

Tokenize text and convert words to numerical sequences.

Use pre-trained GloVe embeddings for vector representation.

Build an LSTM model with input, embedding, LSTM, and dense layers.

Train on training data with categorical cross-entropy loss and Adam optimizer.

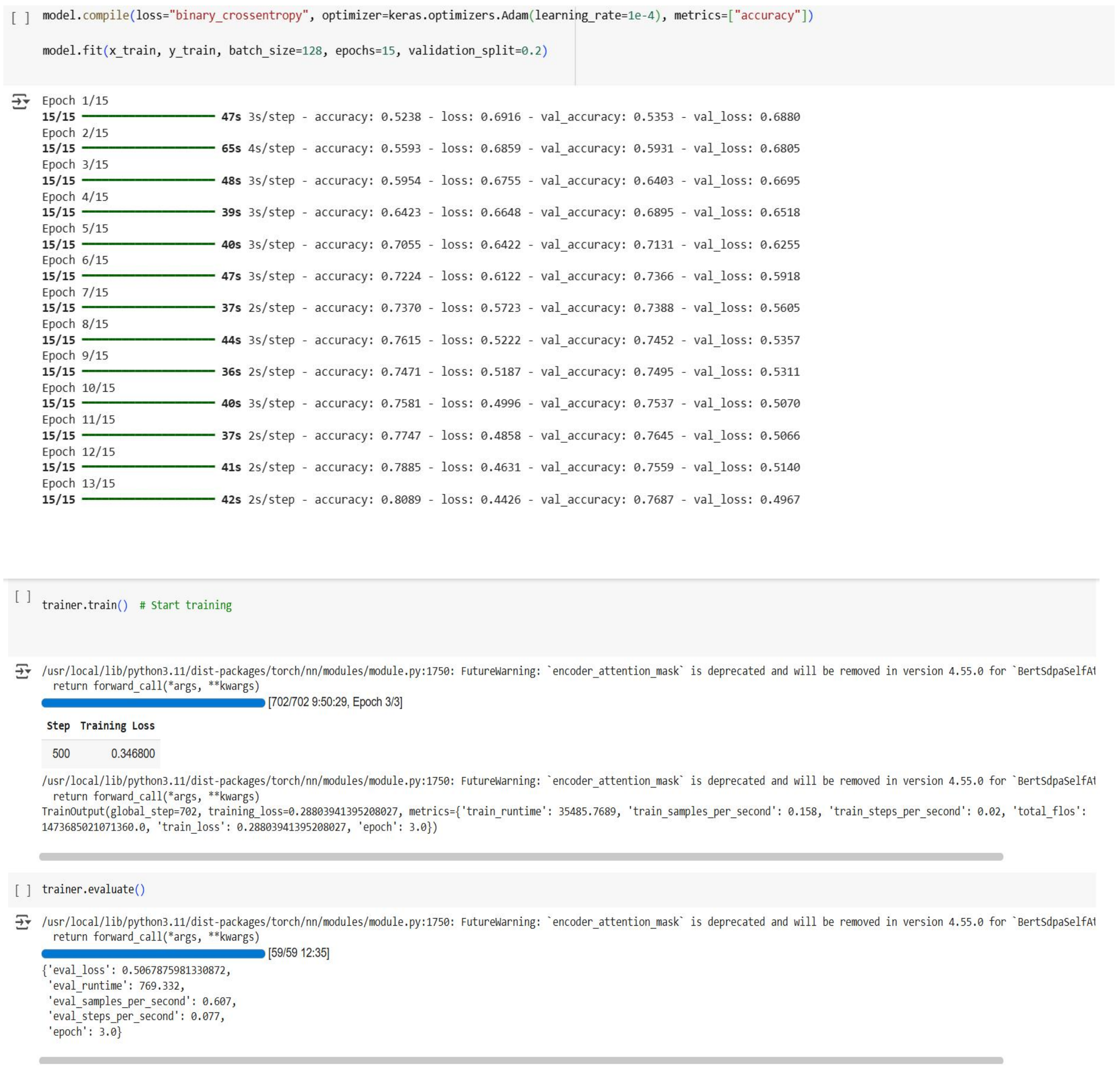
BERT Implementation Steps:

Load a pre-trained BERT model using Hugging Face Transformers.

Tokenize text using BERT’s WordPiece tokenizer.

Fine-tune BERT for binary classification (sarcastic vs. non-sarcastic).

Use Adam optimizer and appropriate learning rate scheduling.



```
[ ] model.compile(loss="binary_crossentropy", optimizer=keras.optimizers.Adam(learning_rate=1e-4), metrics=["accuracy"])

    model.fit(x_train, y_train, batch_size=128, epochs=15, validation_split=0.2)
```

```
Epoch 1/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 47s 3s/step - accuracy: 0.5238 - loss: 0.6916 - val_accuracy: 0.5353 - val_loss: 0.6880
Epoch 2/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 65s 4s/step - accuracy: 0.5593 - loss: 0.6859 - val_accuracy: 0.5931 - val_loss: 0.6805
Epoch 3/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 48s 3s/step - accuracy: 0.5954 - loss: 0.6755 - val_accuracy: 0.6403 - val_loss: 0.6695
Epoch 4/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 39s 3s/step - accuracy: 0.6423 - loss: 0.6648 - val_accuracy: 0.6895 - val_loss: 0.6518
Epoch 5/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 40s 3s/step - accuracy: 0.7055 - loss: 0.6422 - val_accuracy: 0.7131 - val_loss: 0.6255
Epoch 6/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 47s 3s/step - accuracy: 0.7224 - loss: 0.6122 - val_accuracy: 0.7366 - val_loss: 0.5918
Epoch 7/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 37s 2s/step - accuracy: 0.7370 - loss: 0.5723 - val_accuracy: 0.7388 - val_loss: 0.5605
Epoch 8/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 44s 3s/step - accuracy: 0.7615 - loss: 0.5222 - val_accuracy: 0.7452 - val_loss: 0.5357
Epoch 9/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 36s 2s/step - accuracy: 0.7471 - loss: 0.5187 - val_accuracy: 0.7495 - val_loss: 0.5311
Epoch 10/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 40s 3s/step - accuracy: 0.7581 - loss: 0.4996 - val_accuracy: 0.7537 - val_loss: 0.5070
Epoch 11/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 37s 2s/step - accuracy: 0.7747 - loss: 0.4858 - val_accuracy: 0.7645 - val_loss: 0.5066
Epoch 12/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 41s 2s/step - accuracy: 0.7885 - loss: 0.4631 - val_accuracy: 0.7559 - val_loss: 0.5140
Epoch 13/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 42s 2s/step - accuracy: 0.8089 - loss: 0.4426 - val_accuracy: 0.7687 - val_loss: 0.4967
```

```
[ ] trainer.train()  # Start training
```

```
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py:1750: FutureWarning: `encoder_attention_mask` is deprecated and will be removed in version 4.55.0 for `BertSdpaSelfAt
  return forward_call(*args, **kwargs)
[702/702 9:50:29, Epoch 3/3]

Step  Training Loss
500   0.346800

/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py:1750: FutureWarning: `encoder_attention_mask` is deprecated and will be removed in version 4.55.0 for `BertSdpaSelfAt
  return forward_call(*args, **kwargs)
TrainOutput(global_step=702, training_loss=0.28803941395208027, metrics={'train_runtime': 35485.7689, 'train_samples_per_second': 0.158, 'train_steps_per_second': 0.02, 'total_flos':
1473685021071360.0, 'train_loss': 0.28803941395208027, 'epoch': 3.0})
```

```
[ ] trainer.evaluate()
```

```
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py:1750: FutureWarning: `encoder_attention_mask` is deprecated and will be removed in version 4.55.0 for `BertSdpaSelfAt
  return forward_call(*args, **kwargs)
[59/59 12:35]
{'eval_loss': 0.5067875981330872,
 'eval_runtime': 769.332,
 'eval_samples_per_second': 0.607,
 'eval_steps_per_second': 0.077,
 'epoch': 3.0}
```

Fig 4: Validation

CHAPTER 3
MODEL DESCRIPTION

3.1 ML

The sarcasm detection system is developed using **two deep learning-based NLP models**:

Long Short-Term Memory (LSTM)- LSTM is a type of Recurrent Neural Network (RNN) designed to capture long-term dependencies in sequential data. In this project, LSTM processes tokenized text sequences to identify patterns and dependencies that signal sarcasm. Pre-trained **GloVe word embeddings** are used to convert words into dense vector representations, improving the model's ability to understand semantic meaning. The model architecture includes:

Input Layer (tokenized sequences)

Embedding Layer (GloVe vectors)

LSTM Layer (sequence learning)

Dense Layer with sigmoid activation (binary classification: sarcastic / non-sarcastic).

Bidirectional Encoder Representations from Transformers (BERT) -BERT is a transformer-based language model pre-trained on large corpora, designed to understand context in both directions of a sentence. It uses the **attention mechanism** to weigh the importance of each word in relation to others, making it highly effective for sarcasm detection. In this project, the base BERT model is fine-tuned for binary classification. The architecture includes:

BERT Encoder (pre-trained layers)

Fully Connected Layer (classification head)

Softmax Output Layer (sarcastic / non-sarcastic).

3.2 WHY LSTM & BERT

Why LSTM?

Captures **sequential dependencies** in text, making it effective for detecting sarcasm patterns that depend on word order.

Lightweight compared to transformer models, making it suitable for faster training on smaller datasets.

Performs well with **pre-trained embeddings** like GloVe to capture semantic meaning.

Why BERT?

Bidirectional context understanding allows it to interpret sarcasm where meaning is derived from both earlier and later words in a sentence.

Pre-trained on massive datasets, giving it strong generalization capabilities.

The **attention mechanism** helps focus on specific words or phrases that contribute to sarcasm.

By using both models, the project allows for **comparative analysis** between a sequential RNN-based approach (LSTM) and a transformer-based contextual model (BERT),

providing insights into their relative effectiveness in sarcasm detection.

CHAPTER 4
DATA FLOW

The data flow for the sarcasm detection project outlines how data moves through different stages of the system, from collection to prediction.

Data Source - Dataset is collected from Hugging Face Datasets containing labeled sarcastic and non-sarcastic text samples.

Data Preprocessing

Text Cleaning: Removal of unwanted characters, punctuation (if needed), and extra spaces.

Tokenization: Converting sentences into tokens (words/subwords).

LSTM → Standard tokenizer + word index mapping.

BERT → WordPiece tokenizer.

Padding/Truncation: Adjusting sequences to a fixed length.

Embedding: LSTM → GloVe word embeddings.

BERT → Uses its own pre-trained embedding layers.

Model Input Processed and vectorized text is fed into the respective model's input layer.

Model Processing

LSTM Path:

Embedding Layer → LSTM Layer → Dense Output Layer

BERT Path:

BERT Encoder Layers → Classification Head → Output Layer.

Prediction Output

Both models output a probability score for each class (sarcastic or non-sarcastic).

Class with the higher probability is selected as the final prediction.

Evaluation & Comparison

Predictions are compared against actual labels to calculate metrics (Accuracy, Precision, Recall, F1-score).

Performance of LSTM and BERT is compared to identify the more effective model.

CHAPTER 5

PROJECT REQUIREMENTS

Project Requirements

Programming Language:

Python 3.8+

Libraries & Frameworks:

TensorFlow/Keras – For implementing LSTM model

Transformers (Hugging Face) – For implementing BERT model

NumPy & Pandas – Data manipulation and analysis

Matplotlib & Seaborn – Visualization of data and results

Scikit-learn – Evaluation metrics and preprocessing utilities

NLTK / re (Regex) – Text preprocessing and cleaning

GloVe Pre-trained Embeddings – For LSTM model word representation

Development Tools:

Google Colab – For coding and experimentation

Git – Version control

Hugging Face Datasets – Dataset loading

CHAPTER 6
FUTURE SCOPE

The sarcasm detection system developed in this project focuses on short English text and uses LSTM and BERT for binary classification. While it achieves promising results, there is significant scope for enhancement and expansion:

Multilingual Sarcasm Detection
Extend the model to detect sarcasm in multiple languages by using multilingual transformer models like **mBERT** or **XLM-RoBERT.**

Context-Aware Detection
Incorporate conversational context from previous messages or posts to improve sarcasm detection in dialogues and social media threads.

Multi-Modal Sarcasm Detection
Combine text with audio and visual cues (tone, facial expressions) for improved accuracy in sarcasm detection across videos or voice messages.

Real-Time Implementation
Deploy the model into live chat systems, customer service bots, or social media monitoring tools for instant sarcasm detection.

Explainable AI Integration
Use explainability tools (e.g., LIME, SHAP) to show why the model predicted sarcasm, increasing transparency and trust in automated systems.

Handling Code-Mixed Data
Adapt the system for code-mixed language (e.g., Hinglish) where sarcasm often appears in social media contexts.

Improved Data Augmentation
Apply advanced NLP data augmentation techniques (back-translation, synonym replacement, paraphrasing) to improve performance with limited sarcastic samples.

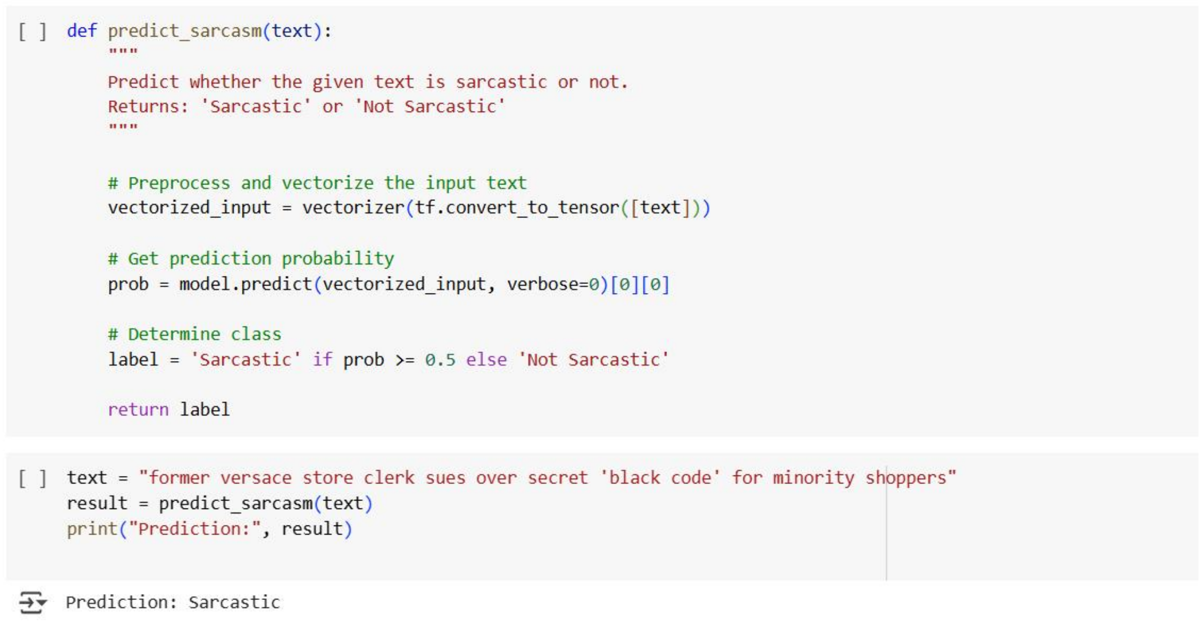
Integration with Sentiment Analysis Systems
Combine sarcasm detection with sentiment analysis pipelines to provide more accurate insights for marketing, customer experience, and brand monitoring.

CHAPTER 7 CONCLUSION

This project successfully demonstrated the use of **Natural Language Processing (NLP)** and deep learning models for detecting sarcasm in text. By implementing and comparing **Long Short-Term Memory (LSTM)** and **Bidirectional Encoder Representations from Transformers (BERT)**, the study highlighted the strengths and limitations of both sequential and transformer-based approaches.

The LSTM model, supported by GloVe embeddings, effectively captured sequential patterns and performed well with relatively lower computational requirements. However, it sometimes struggled with context-dependent sarcasm. On the other hand, BERT leveraged its bidirectional context understanding and attention mechanisms to achieve higher accuracy and better handle subtle sarcastic cues, albeit at the cost of higher computational resources.
Through data preprocessing, model training, evaluation, and comparative analysis, this project provided valuable insights into the challenges of sarcasm detection, such as context dependency, ambiguity, and data imbalance. The findings confirm that context-aware transformer models generally outperform traditional sequence models for this task, making them more suitable for real-world applications.

In conclusion, this work contributes to the growing field of sarcasm detection by presenting a clear comparative study of two powerful deep learning architectures. With further improvements—such as multilingual support, conversational context handling, and real-time deployment- the system can be extended for more robust and practical use cases in sentiment analysis, social media monitoring, and automated customer support.

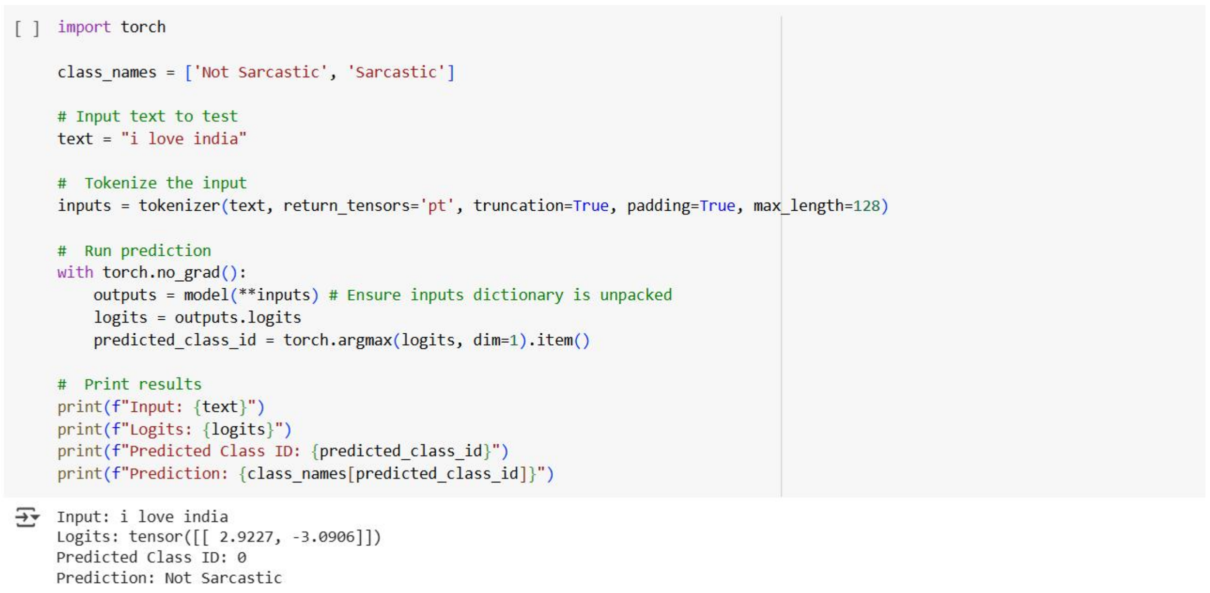


```
[ ] def predict_sarcasm(text):
        """
        Predict whether the given text is sarcastic or not.
        Returns: 'Sarcastic' or 'Not Sarcastic'
        """

        # Preprocess and vectorize the input text
        vectorized_input = vectorizer(tf.convert_to_tensor([text]))

        # Get prediction probability
        prob = model.predict(vectorized_input, verbose=0)[0][0]

        # Determine class
        label = 'Sarcastic' if prob >= 0.5 else 'Not Sarcastic'

        return label
```

```
[ ] text = "former versace store clerk sues over secret 'black code' for minority shoppers"
    result = predict_sarcasm(text)
    print("Prediction:", result)
```

```
Prediction: Sarcastic
```

Fig 5 : Input and output of LSTM



```
[ ] import torch

    class_names = ['Not Sarcastic', 'Sarcastic']

    # Input text to test
    text = "i love india"

    #  Tokenize the input
    inputs = tokenizer(text, return_tensors='pt', truncation=True, padding=True, max_length=128)

    #  Run prediction
    with torch.no_grad():
        outputs = model(**inputs) # Ensure inputs dictionary is unpacked
        logits = outputs.logits
        predicted_class_id = torch.argmax(logits, dim=1).item()

    #  Print results
    print(f"Input: {text}")
    print(f"Logits: {logits}")
    print(f"Predicted Class ID: {predicted_class_id}")
    print(f"Prediction: {class_names[predicted_class_id]}")
```

```
Input: i love india
Logits: tensor([[ 2.9227, -3.0906]])
Predicted Class ID: 0
Prediction: Not Sarcastic
```

Fig 6: Input and Output of BERT

CHAPTER 8
REFERENCES

1. Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735–1780. MIT Press. https://doi.org/10.1162/neco.1997.9.8.1735
2. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, 4171–4186. Association for Computational Linguistics. https://doi.org/10.48550/arXiv.1810.04805
3. Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1532–1543. Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162
4. Wolf, T., Debut, L., Sanh, V., et al. (2020). Transformers: State-of-the-Art Natural Language Processing. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 38–45. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-demos.6
5. Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media, Inc. ISBN: 978-0-596-51649-9.
6. Hugging Face. (2024). Datasets Documentation. Retrieved from https://huggingface.co/docs/datasets
7. Chollet, F. (2015). Keras: Deep Learning for Humans. GitHub Repository. Retrieved from https://keras.io/
8. Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830. Retrieved from https://jmlr.org/papers/v12/pedregosa11a.html