**(CIS – 550) Advance Machine Learning Project Summary**

A comparative analysis of ML models used in Fake-News detection

**Group -26**

**(63) Sinchan Samajdar - 02130429**

**(06) Srinivas Prajwal BR - 02101895**

**(51) Kanishka Patre - 02107654**

Table of Contents:

1. [Summary of problem statement, data and findings 3](#_gjdgxs)
   1. [Problem statement 3](#_30j0zll)
   2. [Data Collection 3](#_1fob9te)
   3. [Data description: 3](#_3znysh7)
   4. [Columns description: 3](#_2et92p0)
   5. [Understanding the dataset in detail 3](#_tyjcwt)
2. [Overview of the final process](#_1t3h5sf) 5
   1. [Text preprocessing:](#_4d34og8) 5
      1. [Text Cleaning Techniques](#_2s8eyo1) 5
      2. [Pre-Processing Techniques 5](#_17dp8vu)
   2. [Model building:](#_1ci93xb) 6
      1. [Gradient Boosting 6](#_3rdcrjn)
      2. [Decision Tree 6](#_26in1rg)
      3. [XGBoost](#_26in1rg) 7
      4. [Random Forest](#_26in1rg) 7
3. [Step-by-step walk through the solution:](#_lnxbz9)  7
   1. [Final Model (LSTM )](#_35nkun2) 7
      1. [Build LSTM Model](#_26in1rg) 7
      2. [Plot the Model 8](#_26in1rg)
      3. [Training Process](#_26in1rg) 9
4. [Model Prediction:](#_lnxbz9)  9
   * 1. [Prediction](#_26in1rg) 9
     2. [Accuracy Evaluation 1](#_26in1rg)0
5. [Comparison to Benchma](#_lnxbz9)rk : 10
6. [Visualizations 1](#_2jxsxqh)1
   1. [Confusion Matrix for Gradient Boosting 1](#_z337ya)1
   2. [Confusion Matrix for Decision Tree 1](#_3whwml4)1
   3. [Confusion Matrix for LSTM 1](#_2bn6wsx)2
   4. [Confusion Matrix for Random Forest 1](#_2bn6wsx)2
   5. [Graph for Model Comparison 1](#_2bn6wsx)3
7. [Implications 1](#_3j2qqm3)3
8. [Limitations 1](#_1y810tw)4
9. [Closing Reflections 1](#_4i7ojhp)4

# 

1. **Summary of problem statement, data and findings:**
   1. Problem statement:

Detecting fake news remains challenging as misinformation tactics evolve. Key questions include building the most accurate detection system and identifying optimal machine learning algorithms and feature sets for this task.

* 1. Data Collection:

Fake News: A system to identify unreliable news articles. (Kaggle)

* 1. Data description:

Datasets includes labeled samples of both real and fake news articles with 5 columns and 20800 rows.

* 1. Columns description:
* **train.csv**: A full training dataset with the following attributes:
* **id**: unique id for a news article
* **title**: the title of a news article
* **author**: author of the news article
* **text**: the text of the article; could be incomplete.
* **label**: a label that marks the article as potentially unreliable.
* 1: unreliable
* 0: reliable
  1. Understanding the dataset in detail:

Received dataset in excel(.xlsx) format used panda’s library to read the dataset. Checking the null values and column data types.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

In summary, these lines of code read a CSV file, preprocess the DataFrame by filling missing values and creating a new column, and then separate the features and target variables while displaying relevant information during the process.

From the shape we could understand that there are a total of 20800 rows and 5 attributes in the dataset.

Here “label” is a target.

# Overview of the final process:

* 1. Text preprocessing:
     1. Text Cleaning Techniques:

* + **Regular Expression for Alphabets:**
    - **m = re.sub("[^a-zA-Z]"," ",text["total"][i])**: This regular expression substitution (**re.sub**) replaces any characters that are not alphabets (letters) with a space. This step removes numbers, punctuation, and other non-alphabetic characters.
  + **Lowercasing:**
    - **m = m.lower()**: Converts all text to lowercase. This is done to ensure uniformity and treat uppercase and lowercase versions of the same word as identical during subsequent processing.
  + **Tokenization and Stemming:**
    - **m = m.split()**: Splits the text into a list of words.
    - **[ps.stem(word) for word in m if not word in stopwords.words('english')]**: Applies stemming using the Porter Stemmer and removes stopwords. Stemming reduces words to their root form, and removing stopwords helps eliminate common and less informative words.
    1. Pre-Processing Techniques:
* **Creating Corpus:**
  + - **clean\_text = " ".join(m)**: Joins the cleaned words back into a sentence. The result is stored in the **clean\_text** variable.
    - **corpus.append(clean\_text)**: Appends each cleaned text to the **corpus** list. The **corpus** contains the cleaned and processed text data.
* **One-Hot Encoding:**
  + - **onehot\_text = [one\_hot(words, VOCAB\_SIZE) for words in clean\_text\_corpus]**: Applies one-hot encoding to each sentence in the **clean\_text\_corpus**. This converts words into unique integer indices based on their frequency in the corpus. The result is a list of lists, where each list represents the one-hot encoding of a sentence.
* **Padding Sequences:**
  + - **padded\_doc = pad\_sequences(onehot\_text, padding="pre", maxlen=25)**: Pads sequences to ensure a consistent length of 25. Sequences shorter than 25 are padded with zeros at the beginning (**padding="pre"**). This step is important for maintaining uniform input shapes when training neural networks.
  1. Model building:

Below are the few Machine learning classifications models we have tried for the given problem statement

* + 1. Gradient Boosting:
* **Description:** Gradient Boosting is an ensemble learning method that combines the predictions of multiple weak learners (typically decision trees) sequentially. It builds trees one at a time, with each tree correcting the errors of the previous one.
* **Implementation:** Utilizes the **GradientBoostingClassifier** from scikit-learn.
* **Accuracy:** Achieved an accuracy of approximately 91.4%.
  + 1. Decision Tree:
* **Description:** A Decision Tree is a tree-like model where an internal node represents a feature, the branch represents a decision rule, and each leaf node represents the outcome. It's a standalone model without ensemble techniques.
* **Implementation:** Utilizes the **DecisionTreeClassifier** from scikit-learn.
* **Accuracy:** Achieved an accuracy of approximately 88.3%.
  + 1. XGBoost:
* **Description:** XGBoost (eXtreme Gradient Boosting) is an efficient and scalable implementation of gradient boosting. It is designed for speed and performance and often outperforms other algorithms. It uses a regularized model with both linear and tree-based learners.
* **Implementation:** Utilizes the **XGBClassifier** from the XGBoost library.
* **Accuracy:** Achieved a high accuracy of approximately 98.0%.
  + 1. Random Forest:
* **Description:** Random Forest is an ensemble learning method that constructs a multitude of decision trees at training time and outputs the class that is the mode of the classes (classification) of the individual trees.
* **Implementation:** Utilizes the **RandomForestClassifier** from scikit-learn.
* **Accuracy:** Achieved an accuracy of approximately 90.9%.

# Step-by-step walk through the solution:

* 1. Final Model (LSTM):

**Build LSTM Model:**

* The **build\_model** function defines an LSTM model using TensorFlow/Keras.
* Model Architecture:
  + Embedding Layer with a vocabulary size of **VOCAB\_SIZE** (embedding dimension: 40) for word representation.
  + Dropout layer (rate=0.3) for regularization.
  + LSTM layer with 100 units to capture sequential patterns in the data.
  + Another Dropout layer.
  + Dense layer with 64 units and ReLU activation.
  + Dropout layer.
  + Final Dense layer with 1 unit and a sigmoid activation for binary classification.
* The model is compiled using the Adam optimizer, binary cross-entropy loss, and binary accuracy as the metric.

A screenshot of a computer program

Description automatically generated

**Plot the Model:**

* **Function:** **tf.keras.utils.plot\_model** generates a visual representation of the LSTM model's architecture.

A diagram of a data flow

Description automatically generated

Training Process:

1. **fit Method:**
   * The **fit** method is used to train the LSTM model on the provided training data (**xtrain** and **ytrain**).
   * The model is trained for 25 epochs with a batch size of 128.
   * 20% of the training data is used for validation (**validation\_split=0.2**).
   * Callbacks are applied during training to implement the specified behaviors.
2. **Monitoring with Callbacks:**
   * The model is trained while being monitored by callbacks.
   * **ModelCheckpoint** ensures that the best model based on validation loss is saved.
   * **EarlyStopping** stops training if there's no improvement after 5 epochs.
   * **ReduceLROnPlateau** adjusts the learning rate if the validation loss plateaus, helping the model converge more effectively.
3. **Saved Model:**
   * The final trained model is saved as "news\_classifier.h5." This saved model can be loaded later for making predictions on new data without retraining.

# Model Prediction:

**Prediction:**

Predicting Labels:

* Using the trained LSTM model (**news\_classifier**), predictions are generated for the test data (**xtest**).
* The **predict** method produces probability scores for each sample in the test set.

Converting Probabilities to Binary Predictions:

* The probability scores are then rounded to the nearest integer using **np.round**. This rounding operation is performed to convert the continuous probability scores into binary predictions (0 or 1).

**Accuracy Evaluation:**

* The **accuracy\_score** function from scikit-learn is employed to calculate the accuracy of the model.
* It compares the predicted binary labels (**predictions**) with the true binary labels of the test data (**ytest**).
* Accuracy is the ratio of correctly predicted instances to the total number of instances in the test set.

Printing Accuracy:

* The computed accuracy is then printed to the console.

Accuracy: 0.9974358974358974

# Comparison to benchmark:

After analyzing all the Machine learning and deep learning models evaluation metrics and by considering accuracy we observed LSTM model has highest accuracy.

| **Model** | **Accuracy** |
| --- | --- |
| Gradient Boosting | 0.9139423076923077 |
| Decision Tree Model | 0.8825320512820513 |
| XGBoost Model | 0.9801282051282051 |
| Random Forest | 0.908974358974359 |
| LSTM | 0.9974358974358974 |

# Visualizations:

* 1. Confusion Matrix for Gradient Boosting

A blue squares with white text

Description automatically generated

* 1. Confusion Matrix for Decision Tree

A blue squares with white text

Description automatically generated

* 1. Confusion Matrix for XGBoost

A blue squares with white text

Description automatically generated

* 1. Confusion Matrix for LSTM

A blue squares with white text

Description automatically generated

* 1. Confusion Matrix for Random Forest

A blue squares with white text

Description automatically generated

* 1. Graph for Model Comparison

A graph of different colored rectangular shapes

Description automatically generated

# Implications:

**Model Performance:**

* The accuracy metric provides an indication of how well the LSTM model can correctly classify news articles as real or fake based on the learned patterns.

**Decision-Making:**

* Depending on the application (e.g., news platforms, fact-checking services), the accuracy of the model influences the reliability of decisions made using its predictions.

**Interpretability:**

* The visual representation of the model architecture (**plot\_model**) can aid in understanding how the different layers of the LSTM contribute to the model's decision-making process.

# Limitations:

* LSTMs assume stationarity in the data, meaning that the underlying statistical properties do not change over time.

**Enhancement**:

1. **Continuous Improvement:**
   * Model performance can be further enhanced through continuous iteration, experimentation with different architectures, hyperparameter tuning, and incorporating more advanced techniques such as attention mechanisms.
2. **Ethical Considerations:**
   * As with any machine learning model, ethical considerations are paramount. Ensuring the model does not inadvertently propagate biases present in the data and maintaining transparency in decision-making are critical aspects.
3. **Monitoring and Adaptation:**
   * Monitoring the model's performance over time is essential. If the distribution of news articles or user behavior changes, the model may need updates or retraining.

# Closing Reflections:

* 1. Model Building:

**Imbalanced Classification Caution:**

* Careful handling is needed for imbalanced classification data to mitigate overfitting risks.

**Data Preprocessing Impact:**

* The choice of vectorization techniques during data preprocessing significantly influences model accuracy.

**Text Data Cleaning:**

* Removal of special characters, numeric numbers, and stop words ensures cleaner data for model training.

**Vectorization Best Practices:**

* Converting text into vector space and separately fitting training and test data helps mitigate overfitting, promoting better generalization.