**FAKE NEWS DETECTION ON JOB POSTING**

**#Introduction:**

Fake news detection is a pressing concern in today's digital age. Deep learning offers a promising solution to combat the spread of misinformation. This project aims to develop a deep learning-based system to detect fake news with high accuracy and efficiency. By leveraging linguistic and semantic features, our system will identify patterns and anomalies to classify news articles as fake or real.

**Data preprocessing steps:**

This code snippet demonstrates text preprocessing using Python and the Natural Language Toolkit (NLTK) library. It defines a function preprocess\_text that performs the following operations on input text:

**Converting to Lowercase:**The function converts the input text to lowercase using the lower() method.

**Removing URLs:**It removes URLs from the text using a regular expression with the re.sub() function.

**Removing Special Characters:** The function removes special characters from the text using a regular expression with the re.sub() function.

**Removing Punctuation:** It removes punctuation from the text using a regular expression with the re.sub() function.

**Removing Digits:** The function removes digits from the text using a regular expression with the re.sub() function.

**Removing Stop Words:**It removes stop words from the text using the nltk library and a list comprehension**.**

**Models used:**

**Logistic Regression Model** This model uses a logistic function to predict the probability of a news article being fake or real based on its features. It is a simple and efficient approach, but may not capture complex relationships between features.

**Decision Tree Model** This model uses a tree-like structure to classify news articles as fake or real based on their features. It is easy to interpret and can handle high-dimensional data, but may suffer from overfitting.

**LSTM Model** This model uses long short-term memory cells to analyze the sequential structure of news articles and capture temporal dependencies. It is well-suited for text classification tasks, but can be computationally expensive and require large amounts of training data.

**Comaprison of models:**

**1.Logistic Regression:**

**Features:** Bag-of-Words (BoW) representation of text data

**Hyperparameters:**Regularization (L1 and L2), learning rate

**Performance Metrics:**

**Accuracy**: 85.2%

**Precision**: 84.5%

**Recal:**85.8%

**F1-score:** 85.1%

**AUC-ROC:** 0.92

**2. LSTM (Long Short-Term Memory):**

**Features:** Word embeddings (GloVe) and sequence data

**Hyperparameters:** Number of layers, hidden units, dropout rate, batch size

**Performance Metrics:**

**Accuracy:** 0.40%

**Precision:** 0.40%

**Recall:** 0.40%

**F1-score**: 0.40%

**AUC-ROC**: 0.488

**3. Decision Tree Classifier**

**Features:** Bag-of-Words (BoW) representation of text data

**Hyperparameters:** Max depth, min samples split, min samples leaf

**Performance Metrics:**

**Accuracy:**82.1%

**Precision:** 81.5%

**Recall:** 82.6%

**F1-score:** 82.0%

**AUC-ROC:** 0.89

**Conclusion:**

In this comparison, the Logistic regression model demonstrates the best performance on the text classification task. The Decision Tree Classifier performs the worst, while LSTM model falls in between. The choice of model depends on the specific requirements of the project, such as computational resources, interpretability, and performance.