Sentiment Analysis Project Report

1. Introduction

This project focuses on sentiment analysis using machine learning and deep learning models. Sentiment analysis is an essential task in Natural Language Processing (NLP) that involves determining the sentiment expressed in textual data. The primary objective of this project is to classify user reviews of an app into three sentiment categories: Positive, Negative, and Neutral. I have trained and evaluated five different models to compare their effectiveness in sentiment classification.

Analyzing customer sentiment is crucial for businesses and app developers as it helps in understanding user feedback, improving products, and making data-driven decisions. By applying machine learning techniques, we can automate the process of classifying large volumes of reviews efficiently.

2. Data Preprocessing

To ensure that the dataset is properly formatted for machine learning models, several preprocessing steps were carried out:

- **Lowercasing**: All text was converted to lowercase to maintain uniformity.
- **Removing special characters and numbers**: Punctuation and numbers were eliminated to focus on the meaningful words.
- **Tokenization**: The text was broken down into individual words for analysis.

- **Stop word removal**: Common words that do not add significant meaning (e.g., "the", "is", "and") were removed to improve model performance.

```
# Function for Text Preprocessing
def clean_text(text):
    if not isinstance(text, str):
        return ""
    text = text.lower()
    text = re.sub(r'http\S+', '', text)  # Curling out the URL
    text = re.sub(r'[^a-zA-Z\s]', '', text)  # Erasing special
    text = text.strip()
    return text

df["Cleaned Review"] = df["Review"].apply(clean_text)
```

- **Vectorization**: Two methods were used:
    - TF-IDF (Term Frequency-Inverse Document Frequency) for traditional machine learning models.
    -

```
# Vectorization (TF-IDF)

 # Increased max features for better text representation
tfidf = TfidfVectorizer(max_features=7000)
X = tfidf.fit_transform(df["Cleaned Review"]).toarray()
```

    Word embeddings for deep learning models, converting words into numerical representations.

- **Data Splitting**: The dataset was split into an **80% training set** and a **20% testing set** to evaluate model performance on unseen data.

```
# Data Splitting
X_train, X_test, y_train, y_test = train_test_split(X, y, test
random_state=42, stratify=y)

print("Data preprocessing complete!")
```

```
Data preprocessing complete!
```

These preprocessing steps were essential in ensuring that the models could learn meaningful patterns from the textual data rather than noise.

3. Model Training

Several machine learning and deep learning models were trained to classify the sentiments:

- **Logistic Regression**: A linear classification model that predicts sentiment based on weighted word features.
- **Naive Bayes**: A probabilistic classifier based on Bayes' theorem, often used for text classification tasks.
- **Support Vector Machine (SVM)**: A model that finds an optimal hyperplane to separate sentiment classes.
- **Random Forest Classifier**: An ensemble learning method that builds multiple decision trees to improve accuracy.
- **LSTM (Long Short-Term Memory)**: A type of recurrent neural network (RNN) designed to capture sequential dependencies in text, making it effective for NLP tasks.

Each model was trained on the preprocessed dataset, and hyperparameters were adjusted to enhance performance.

4. Model Evaluation

The models were evaluated using accuracy scores based on predictions on the test dataset. The results are as follows:

Model	Accuracy
Logistic Regression	62 %
Naive Bayes	69 %
SVM	39%
Random Forest	65 %
LSTM	59.76%

From the evaluation, **Naive Bayes** achieved the highest accuracy of **69%**, making it the best-performing model., showing their reliability in handling textual sentiment classification. **Random Forest followed with 65%**, while Logistic Regression and LSTM had **61% and 60%**, respectively., demonstrating their effectiveness in traditional NLP tasks. **SVM had the lowest accuracy at 38%**, indicating its struggles with this dataset., which could be attributed to the dataset's small size and the limited number of training epochs.

Although LSTM models are generally well-suited for sequential text classification, their performance highly depends on the amount of training data and hyperparameter tuning. On the other hand, Logistic Regression and Random Forest performed well due to their ability to capture relationships between words and sentiment labels efficiently.

A bar chart was generated to visualize the model performance comparison, providing insights into which models performed best for sentiment analysis tasks given the dataset constraints. **SVM had significantly lower accuracy (38%)**, while Random Forest performed well with 65%. but required more training time compared to other models.

5. Conclusion

This project demonstrated the effectiveness of different machine learning and deep learning models for sentiment analysis. The key findings include:

- **Naive Bayes** achieved the highest accuracy of **69%**, making it the most effective model for sentiment classification., making them strong choices for text classification tasks.
- **Random Forest followed with 65%**, while Logistic Regression and LSTM had **61% and 60%**, respectively. accuracy, showing their effectiveness for sentiment classification.
- **SVM had the lowest accuracy at 38%**, indicating its struggles with this dataset., possibly due to the small dataset size and limited training epochs.
- Preprocessing steps such as tokenization, vectorization, and text cleaning significantly influenced model performance.