Report: NLP Sentiment Analysis Model

1. Introduction:

The aim of this project is to build a sentiment analysis model using Natural Language Processing (NLP). The model classifies text into three categories: happiness, angriness, and sadness. The project uses various NLP techniques, including text cleaning, feature transformation, and machine learning models.

2. Design Choices:

2.1. Data Loading and Labelling:

Three datasets, representing different sentiments (angry, happy, sad), are loaded and concatenated.

A label mapping is created to convert sentiment labels into numerical values.

2.2. Data Preprocessing:

Text data is cleaned using functions like `remove_stopwords` and `clean_Data` to remove unnecessary information and reduce noise.

Duplicate entries are identified and removed.

The length of the content is calculated and analyzed for each sentiment label before and after cleaning the data.Character count,word count,word density are also evaluated on the content.

2.3. Feature Transformation:

Two types of feature transformation are explored: Count Vectorization and TF-IDF Vectorization.

Vectorization is performed on the cleaned content to convert text into numerical features for model training.

2.4. Model Selection:

Five classifiers are chosen for evaluation: Logistic Regression, Naive Bayes, Random Forest, Gradient Boosting and Decision Tree Classifier.

The models are trained on both Count Vectorization and TF-IDF Vectorization.

3. Performance Evaluation:

3.1. Model Training and Evaluation:

Models are trained and evaluated using accuracy and area under the ROC curve (AUC) as performance metrics.

Performance is compared for both Count Vectorization and TF-IDF Vectorization.

3 fold Cross validation is performed during training models.

3.2. Results:

Random Forest outperforms all the other models in both accuracy and AUC. TF-IDF Vectorization generally provides better results compared to Count Vectorization.

4. Discussion of Future Work:

4.1. Model Improvement:

Fine-tune hyperparameters of the chosen models to potentially enhance performance. Explore more advanced algorithms and ensemble methods for improved accuracy.

4.2. Feature Engineering:

Experiment with different text preprocessing techniques and feature engineering methods to capture more nuanced information from the text.

5. Source Code:

The provided source code encompasses the entire pipeline, including data loading, preprocessing, feature transformation, model training, and evaluation. Key libraries such as

```
# Importing essential libraries and functions import pandas as pd
import numpy as np
import nltk
import re
import string
nltk.download("stopwords")
nltk.download("punkt_tab")
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

```
#Visualisation Library
import matplotlib.pyplot as plt
import seaborn as sns
! pip install wordcloud
from wordcloud import WordCloud
```

```
# Feature Transformation Library
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
#Classification Report and model evaluation
from sklearn.metrics import roc_auc_score, classification_report
 from sklearn.metrics import accuracy_score
```

Conclusion:

The Intensity analysis model demonstrates promising performance, with Logistic Regression and Random Forest emerging as strong contenders. Future work involves refining the model, and exploring advanced NLP techniques to further enhance accuracy. The provided source code serves as a comprehensive guide for building and evaluating Intensity analysis models.