Model Optimization and Tuning Phase Template

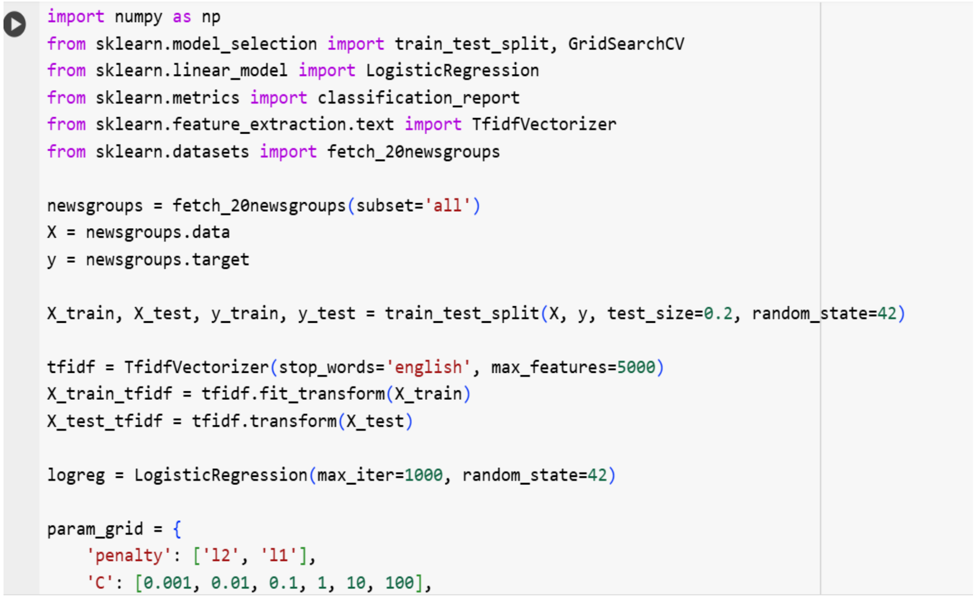
Date	15 March 2024
Team ID	SWTID1728136330
Project Title	Fake News Analysis in Social Media Using NLP
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining neural network models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

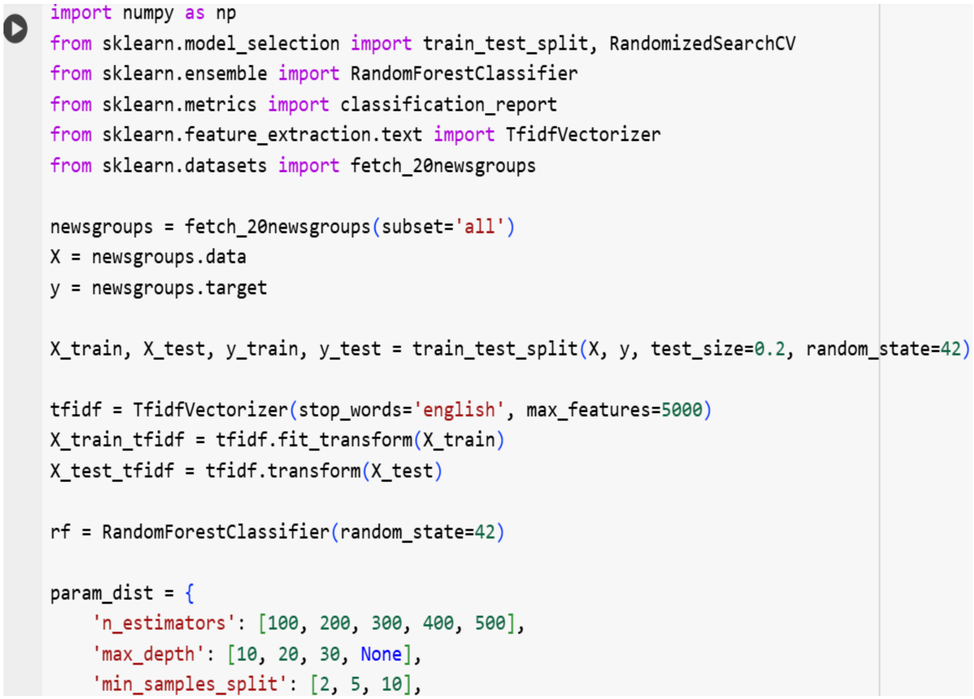
Hyperparameter Tuning Documentation (8 Marks):

<table>
<tr><th>Model</th><th>Tuned Hyperparameters</th></tr>
<tr><td>Model 1</td><td>



```
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.datasets import fetch_20newsgroups

newsgroups = fetch_20newsgroups(subset='all')
X = newsgroups.data
y = newsgroups.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

tfidf = TfidfVectorizer(stop_words='english', max_features=5000)
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)

logreg = LogisticRegression(max_iter=1000, random_state=42)

param_grid = {
    'penalty': ['l2', 'l1'],
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
```

</td></tr>
</table>

<table>
<tr><td>Model 2</td><td>
<pre>
import numpy as np
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.datasets import fetch_20newsgroups

newsgroups = fetch_20newsgroups(subset='all')
X = newsgroups.data
y = newsgroups.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

tfidf = TfidfVectorizer(stop_words='english', max_features=5000)
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)

rf = RandomForestClassifier(random_state=42)

param_dist = {
    'n_estimators': [100, 200, 300, 400, 500],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10],
</pre></td></tr>
<tr><td>…</td><td>…</td></tr>
</table>

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Model 1 (or other)	1.Logistic Regression (Hyperparam1)

	Why Chosen: Simple, interpretable, and effective for binary classification tasks like fake vs. real news. Advantages: Fast, efficient, and easy to deploy in real-time applications. The model is also interpretable, helping explain which features (words) are most important. Key Hyperparameters: Regularization (`l2`), strength (`C`), and solver (`liblinear`) were tuned for optimal performance. 2.Random Forest (Hyperparam2) Why Chosen: Handles complex, non-linear relationships and gives higher accuracy than Logistic Regression. Advantages: Robust to overfitting and effective with large, high-dimensional datasets. It captures complex patterns better. Key Hyperparameters: Number of trees (`n_estimators`), tree depth (`max_depth`), and minimum samples to split or leaf (`min_samples_split`, `min_samples_leaf`) were optimized. Conclusion Logistic Regression offers simplicity and speed, making it ideal for quick, interpretable predictions. Random Forest provides higher accuracy and robustness for more complex patterns, making it ideal for capturing non-linear relationships in text data.