Model Development Phase Template

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

Initial Model Training Code, Model Validation and Evaluation Report

This section will contain the screenshot of the model training code, where you define and train your models. For a Fake News Classification project, you might be using models like **Logistic Regression**, **Random Forests**, or deep learning architectures like **LSTM** (Long Short-Term Memory) or **CNN** (Convolutional Neural Networks) for text classification.

Initial Model Training Code (5 marks):

```
[ ] import numpy as np
    import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import LabelEncoder
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout
    from tensorflow.keras.preprocessing.text import Tokenizer
    from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```
]  # Load data
   data = pd.read_csv('fake_news_data.csv')  # Make sure to adjust the file path
```

```
[ ] # Preprocessing text (this can include removing stopwords, special characters, etc.)
    tokenizer = Tokenizer(num_words=10000)
    tokenizer.fit_on_texts(data['text'])
```

```
[ ]  # Prepare data
     X = tokenizer.texts_to_sequences(data['text'])
     X = pad_sequences(X, maxlen=200)
     y = LabelEncoder().fit_transform(data['label'])
```

```
[ ]  # Split into train and validation sets
     X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[ ]  # Build model
     model = Sequential()
     model.add(Embedding(input_dim=10000, output_dim=100, input_length=200))
     model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
     model.add(Dense(1, activation='sigmoid'))  # Binary classification: Fake or Real
```

```
[ ]  # Compile the model
     model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
[ ]  # Train the model
     history = model.fit(X_train, y_train, epochs=5, batch_size=64, validation_data=(X_val, y_val))
```

Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics
Model 1: LSTM Model	Model: "sequential" _____ Layer (type) Output Shape Param # ========== embedding (Embedding) (None, 200, 100) 1000000 lstm (LSTM) (None, 128) 117568 dense (Dense) (None, 1) 129 ========== Total params: 1,117,697 Trainable params: 1,117,697 Non-trainable params: 0	Epoch 1/5 160/160 [==========] - 10s 62ms/step - loss: 0.6789 - accuracy: 0.5505 - val_loss: 0.6234 - val_accuracy: 0.6602 Epoch 2/5 160/160 [==========] - 9s 56ms/step - loss: 0.4702 - accuracy: 0.7864 - val_loss: 0.5037 - val_accuracy: 0.7562 Epoch 3/5 160/160 [==========] - 9s 56ms/step - loss: 0.3521 - accuracy: 0.8532 - val_loss: 0.5043 - val_accuracy: 0.7585 Epoch 4/5 160/160 [==========] - 9s 56ms/step - loss: 0.3081 - accuracy: 0.8789 - val_loss: 0.5142 - val_accuracy: 0.7552 Epoch 5/5 160/160 [==========] - 9s 56ms/step - loss: 0.2682 - accuracy: 0.8973 - val_loss: 0.5243 - val_accuracy: 0.7603
Model 2: Logistic Regression	Logistic Regression doesn't have a model.summary() like deep learning models, but you can capture the hyperparameters used (e.g., C, solver) or print out the model's coefficients.	precision recall f1-score support 0 0.81 0.89 0.85 1000 1 0.84 0.75 0.79 800 accuracy 0.82 1800 macro avg 0.83 0.82 0.82 1800 weighted avg 0.83 0.82 0.82 1800