Task

Evaluate different machine learning models, including FastText, TF-IDF with traditional ML, and Sentence Transformers with a classifier, for email classification (Spam, Promotions, Support, Personal) using the "emails.csv" dataset. Compare their performance and develop a function to automate predictions with the best-performing model.

Project Setup and Data Preparation

Subtask:

Install all necessary Python libraries, load the provided email dataset, and split it into training and testing sets for model development and evaluation.

```
!pip install fasttext
```

```
Requirement already satisfied: fasttext in /usr/local/lib/python3.12/dist-packages (0.9.3)
Requirement already satisfied: pybind11>=2.2 in /usr/local/lib/python3.12/dist-packages (from fasttext) (3.0.1
Requirement already satisfied: setuptools>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from fasttext) (7
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (from fasttext) (2.0.2)
```

```
import pandas as pd

df = pd.read_csv('emails.csv')
print("First 5 rows of the DataFrame:")
print(df.head())
print("\nColumn names of the DataFrame:")
print(df.columns)
```

```
First 5 rows of the DataFrame:
                              text        label
0           Win a free iPhone now         Spam
1      Limited time discount offer  Promotions
2        My internet is not working     Support
3      Hey are we meeting tomorrow?    Personal
4  Reset your password immediately        Spam

Column names of the DataFrame:
Index(['text', 'label'], dtype='object')
```

```
X = df['text']
y = df['label']

print("Features (X) head:")
print(X.head())
print("\nTarget (y) head:")
print(y.head())
```

```
Features (X) head:
0              Win a free iPhone now
1        Limited time discount offer
2          My internet is not working
3       Hey are we meeting tomorrow?
4    Reset your password immediately
Name: text, dtype: object
```

```
Target (y) head:
0          Spam
1    Promotions
2       Support
3      Personal
4          Spam
Name: label, dtype: object
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.25,
    random_state=42
)

print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")
```

```
X_train shape: (6,)
X_test shape: (2,)
y_train shape: (6,)
y_test shape: (2,)
```

FastText Model for Email Classification

Subtask:

Train a FastText model using the prepared training data and then evaluate its performance on the unseen test set to classify emails.

```
with open("fasttext_train.txt", "w") as f:
    for text, label in zip(X_train, y_train):
        f.write(f"__label__{label} {text}\n")

print("fasttext_train.txt created")

with open("fasttext_test.txt", "w") as f:
    for text, label in zip(X_test, y_test):
        f.write(f"__label__{label} {text}\n")

print("fasttext_test.txt created")
```

```
fasttext_train.txt created
fasttext_test.txt created
```

```
import fasttext

model_ft = fasttext.train_supervised(
    input="fasttext_train.txt",
    epoch=25,
    lr=1.0,
    wordNgrams=2
)

print("FastText model trained successfully.")

# Evaluate the model
```

```
result = model_ft.test("fasttext_test.txt")

print("\nFastText Model Evaluation on Test Set:")
print(f"Precision: {result[1]:.4f}")
print(f"Recall: {result[2]:.4f}")
print(f"F1-Score: {result[1]*2*result[2]/(result[1]+result[2]):.4f}")
```

```
FastText model trained successfully.

FastText Model Evaluation on Test Set:
Precision: nan
Recall: nan
F1-Score: nan
```

FastText Evaluation Metrics Explanation

The FastText model evaluation resulted in `nan` values for Precision, Recall, and F1-Score. This is primarily due to the extremely small size and specific composition of our test set. With only two test samples, both belonging to the 'Promotions' class, the calculation of these metrics can become undefined, especially if the model does not predict any of these as 'Promotions' or if the internal averaging mechanisms of FastText encounter division by zero (e.g., no true positives or predicted positives for a given class).

While the model training and evaluation steps were executed as per the instructions, the interpretability of these metrics is limited by the current dataset's size and distribution. For more robust evaluation metrics, a larger and more diverse test set would be necessary.

TF-IDF Vectorization with Traditional ML

Subtask:

Apply TF-IDF (Term Frequency-Inverse Document Frequency) to vectorize the email text. Then, train a traditional machine learning classifier (e.g., Logistic Regression or Support Vector Machine) using these TF-IDF features and evaluate its performance.

Reasoning: The previous FastText model evaluation yielded `nan` values, indicating an issue with the evaluation on the small dataset. This could be due to having too few samples in the test set after splitting, leading to some labels not being present in the test set or insufficient data for certain metrics. As per the subtask, I will now proceed with TF-IDF vectorization and traditional ML. I'll start by importing `TfidfVectorizer`, initializing it, fitting it on the training data, and then transforming both training and test data.

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Initialize TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()

# Fit and transform X_train
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)

# Transform X_test
X_test_tfidf = tfidf_vectorizer.transform(X_test)

print("TF-IDF vectorization complete.")
print(f"Shape of X_train_tfidf: {X_train_tfidf.shape}")
print(f"Shape of X_test_tfidf: {X_test_tfidf.shape}")
```

```
TF-IDF vectorization complete.
Shape of X_train_tfidf: (6, 26)
Shape of X_test_tfidf: (2, 26)
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

# Initialize LogisticRegression model
logistic_model = LogisticRegression(random_state=42)

# Train the model
logistic_model.fit(X_train_tfidf, y_train)

print("Logistic Regression model trained successfully.")

# Make predictions on the test set
y_pred = logistic_model.predict(X_test_tfidf)

# Print evaluation metrics
print("\nLogistic Regression Model Evaluation on Test Set:")
print("Classification Report:")
print(classification_report(y_test, y_pred, zero_division=0))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Logistic Regression model trained successfully.

Logistic Regression Model Evaluation on Test Set:
Classification Report:
              precision    recall  f1-score   support

  Promotions       0.00      0.00      0.00       2.0
        Spam       0.00      0.00      0.00       0.0

    accuracy                           0.00       2.0
   macro avg       0.00      0.00      0.00       2.0
weighted avg       0.00      0.00      0.00       2.0


Confusion Matrix:
[[0 2]
 [0 0]]
```

TF-IDF with Logistic Regression Evaluation Metrics Explanation

The Logistic Regression model, after being trained on TF-IDF features, showed a performance of 0.00 for precision, recall, and F1-score. The confusion matrix further indicates that the model was unable to correctly classify the test samples. This poor performance is expected given the extremely limited size of our dataset. With only 6 samples in the training set and 2 in the test set, the model lacks sufficient data to learn meaningful patterns and generalize. Specifically, in the test set, both samples belong to the 'Promotions' class, and the model seems to have failed to predict these correctly, possibly defaulting to another class or struggling with the sparse representation from TF-IDF on such little data.

While the steps for TF-IDF vectorization and Logistic Regression training and evaluation were successfully executed, the results highlight the critical importance of having a sufficiently large and balanced dataset for effective machine learning model training and evaluation.

Embedding-based Classification using Sentence Transformers

Subtask:

Utilize pre-trained models from the `sentence-transformers` library to generate powerful semantic embeddings for the email text. Train a separate classifier on these embeddings and evaluate its performance.

Reasoning: The next step is to import the necessary libraries, including `SentenceTransformer` for generating embeddings and `LogisticRegression`, `classification_report`, and `confusion_matrix` for model training and evaluation, as specified in the instructions.

```
from sentence_transformers import SentenceTransformer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

print("Libraries imported successfully.")
```

```
Libraries imported successfully.
```

```
model_st = SentenceTransformer('all-MiniLM-L6-v2')

X_train_embeddings = model_st.encode(X_train.tolist())
X_test_embeddings = model_st.encode(X_test.tolist())

print("Sentence embeddings generated successfully.")
print(f"Shape of X_train_embeddings: {X_train_embeddings.shape}")
print(f"Shape of X_test_embeddings: {X_test_embeddings.shape}")
```

```
Loading weights: 100%          103/103 [00:00<00:00, 259.21it/s, Materializing param=pooler.dense.weight]
BertModel LOAD REPORT from: sentence-transformers/all-MiniLM-L6-v2
Key                     | Status     |  |
------------------------+------------+--+-
embeddings.position_ids | UNEXPECTED |  |

Notes:
- UNEXPECTED    :can be ignored when loading from different task/architecture; not ok if you expect identi
Sentence embeddings generated successfully.
Shape of X_train_embeddings: (6, 384)
Shape of X_test_embeddings: (2, 384)
```

```
logistic_model_st = LogisticRegression(random_state=42)

# Train the model on sentence embeddings
logistic_model_st.fit(X_train_embeddings, y_train)

print("Logistic Regression model trained on Sentence Transformer embeddings successfully.")

# Make predictions on the test set
y_pred_st = logistic_model_st.predict(X_test_embeddings)

# Print evaluation metrics
print("\nLogistic Regression Model Evaluation on Test Set (Sentence Transformer Embeddings):")
print("Classification Report:")
print(classification_report(y_test, y_pred_st, zero_division=0))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred_st))
```

```
Logistic Regression model trained on Sentence Transformer embeddings successfully.

Logistic Regression Model Evaluation on Test Set (Sentence Transformer Embeddings):
Classification Report:
              precision    recall  f1-score   support

  Promotions       0.00      0.00      0.00       2.0
        Spam       0.00      0.00      0.00       0.0
     Support       0.00      0.00      0.00       0.0

    accuracy                           0.00       2.0
   macro avg       0.00      0.00      0.00       2.0
weighted avg       0.00      0.00      0.00       2.0


Confusion Matrix:
[[0 1 1]
 [0 0 0]
 [0 0 0]]
```

```
def predict_email_category(email_text):
    """
    Predicts the category of an email using the Sentence Transformer-based Logistic Regression model.

    Args:
        email_text (str): The text content of the email.

    Returns:
        str: The predicted category of the email.
    """
    # Generate embedding for the new email text
    email_embedding = model_st.encode([email_text])

    # Predict the label using the trained logistic regression model
    prediction = logistic_model_st.predict(email_embedding)

    return prediction[0]

print("Prediction function created. Now let's test it with an example.")

# Example usage
new_email = "You won a lottery! Claim your prize now."
predicted_category = predict_email_category(new_email)
print(f"\nNew email: '{new_email}'")
print(f"Predicted category: {predicted_category}")

new_email_2 = "Just wanted to check in and see how you're doing."
predicted_category_2 = predict_email_category(new_email_2)
print(f"\nNew email: '{new_email_2}'")
print(f"Predicted category: {predicted_category_2}")
```

```
Prediction function created. Now let's test it with an example.

New email: 'You won a lottery! Claim your prize now.'
Predicted category: Spam

New email: 'Just wanted to check in and see how you're doing.'
Predicted category: Personal
```

Comparative Analysis of Models

Subtask:

Compare the performance of all trained models (FastText, TF-IDF + Classifier, and Sentence Transformer-based Classifier) using key metrics like F1-score and confusion matrices to identify the most effective approach.

FastText Model Performance Review

The FastText model evaluation resulted in `nan` values for Precision, Recall, and F1-Score. As previously explained, this is primarily due to the extremely small size and specific composition of our test set. With only two test samples, both belonging to the 'Promotions' class, the calculation of these metrics can become undefined, especially if the model does not predict any of these as 'Promotions' or if the internal averaging mechanisms of FastText encounter division by zero. This makes a direct quantitative comparison with other models using these metrics challenging for FastText in this specific scenario.

TF-IDF with Logistic Regression Model Performance Review

The TF-IDF with Logistic Regression model showed a performance of 0.00 for precision, recall, and F1-score across all classes. The classification report indicated that for the 'Promotions' class (which had 2 support samples), the model achieved 0.00 in all metrics. The confusion matrix also showed that the model failed to correctly classify any of the test samples, with predictions potentially defaulting to a class other than 'Promotions' or 'Spam' (although the report only shows two classes). This result is highly indicative of the model's inability to learn meaningful patterns from the extremely limited training data and generalize to the test set, especially with a sparse representation from TF-IDF on such little data.

Sentence Transformer-based Logistic Regression Model Performance Review

The Sentence Transformer-based Logistic Regression model also exhibited a performance of 0.00 for precision, recall, and F1-score across all classes. The classification report indicated that for the 'Promotions' class (which had 2 support samples), the model achieved 0.00 in all metrics. The confusion matrix further illustrated the model's inability to correctly classify the test samples. The test set contained two samples, both labeled 'Promotions'. The model incorrectly predicted one as 'Spam' and the other as 'Support'. This outcome, similar to the other models, underscores the profound limitations imposed by the minuscule dataset size. While sentence embeddings are generally powerful, their effectiveness is severely hampered when there is insufficient data to train the subsequent classifier, preventing it from learning any meaningful decision boundaries.

Sentence Transformer-based Logistic Regression Model Performance Review

The Sentence Transformer-based Logistic Regression model also exhibited a performance of 0.00 for precision, recall, and F1-score across all classes. The classification report indicated that for the 'Promotions' class (which had 2 support samples), the model achieved 0.00 in all metrics. The confusion matrix further illustrated the model's inability to correctly classify the test samples. The test set contained two samples, both labeled 'Promotions'. The model incorrectly predicted one as 'Spam' and the other as 'Support'. This outcome, similar to the other models, underscores the profound limitations imposed by the minuscule dataset size. While sentence embeddings are generally powerful,

their effectiveness is severely hampered when there is insufficient data to train the subsequent classifier, preventing it from learning any meaningful decision boundaries.

Summary of Model Performance and Limitations

Across all three models—FastText, TF-IDF with Logistic Regression, and Sentence Transformer-based Logistic Regression—the observed performance was uniformly poor, primarily due to the extremely limited size of the dataset. Each model struggled to generate meaningful classifications, resulting in 0.00 for precision, recall, and F1-score, or `nan` values where calculations were undefined due to insufficient data points.

- **FastText:** The evaluation yielded `nan` values, which is a direct consequence of the test set having only two samples, both belonging to the 'Promotions' class. With such a restricted and imbalanced test set, the model's performance could not be reliably quantified, and a direct comparison using these metrics is not possible.
- **TF-IDF with Logistic Regression:** This model also performed at 0.00 across all metrics. The sparse nature of TF-IDF features, combined with only 6 training samples and 2 test samples (both 'Promotions'), meant the model couldn't learn any effective patterns. The confusion matrix indicated complete misclassification, highlighting the severe data scarcity issue.
- **Sentence Transformer-based Logistic Regression:** Despite using powerful pre-trained embeddings, this model similarly registered 0.00 for all metrics. The two test samples ('Promotions') were incorrectly predicted as 'Spam' and 'Support'. While Sentence Transformers generate rich semantic representations, the subsequent Logistic Regression classifier lacked sufficient data to establish robust decision boundaries between classes, leading to random or biased predictions.

Overall Conclusion on Performance:

All models demonstrated an inability to perform effectively on this dataset. It is not possible to identify a 'best' performing model or even a slight indication of better potential given these results. The primary and overriding factor for this universal failure is the **critically small dataset size**. With only 8 total email entries, and an even smaller split for training (6 samples) and testing (2 samples), none of the models had adequate data to learn, generalize, or be meaningfully evaluated. The `nan` and 0.00 metric values are direct symptoms of this data limitation, rendering any detailed comparative analysis of their inherent strengths or weaknesses impossible under these conditions.

Automating Predictions

Subtask:

Develop a function that can take new, unseen email text and automatically predict its category (Spam, Promotions, Support, Personal) using one of the trained models for demonstration, acknowledging the limitations of model selection due to small dataset size.

Summary:

Q&A

- **Which machine learning model performed best for email classification?** It was not possible to identify a best-performing model among FastText, TF-IDF with Logistic Regression, and Sentence Transformers with Logistic Regression. All models exhibited uniformly poor performance, with metrics often resulting in 0.00 or `nan`, primarily due to the critically small dataset size.

Data Analysis Key Findings

- The initial dataset contained 8 email entries, which were split into 6 samples for training and 2 samples for testing.
- **FastText Model**: The model evaluation resulted in `nan` values for Precision, Recall, and F1-Score. This was attributed to the extremely small test set (only 2 samples, both belonging to the 'Promotions' class), rendering reliable quantitative comparison impossible.
- **TF-IDF with Logistic Regression**: This model achieved 0.00 for precision, recall, and F1-score across all classes. The confusion matrix confirmed that the model failed to correctly classify any of the test samples, indicating an inability to learn meaningful patterns from the severely limited training data.
- **Sentence Transformer-based Logistic Regression**: Similar to the TF-IDF model, this approach also yielded 0.00 for precision, recall, and F1-score across all classes. The two test samples (both 'Promotions') were incorrectly predicted as 'Spam' and 'Support', highlighting a complete failure in classification despite using pre-trained semantic embeddings.
- **Overall Model Performance**: All three evaluated models demonstrated an inability to perform effectively on the dataset. The primary and overriding factor for this universal failure was the critically small dataset size, which prevented models from learning, generalizing, or being meaningfully evaluated.

Insights or Next Steps

- The primary next step is to **acquire a significantly larger and more diverse dataset** to enable meaningful model training, evaluation, and comparative analysis. Without sufficient data, model performance metrics are unreliable, and the selection of an effective classification approach remains unfounded.
- It is crucial to recognize that the effectiveness of even advanced models, such as those leveraging Sentence Transformers, is severely constrained by data scarcity. Robust model evaluation and comparison require adequate data to establish meaningful patterns and test generalization capabilities.