Q2 - Decision Tree Classifier (EMAIL Dataset) - Detailed Answers

Problem statement

Use Decision Tree classification algorithm to build a classifier for EMAIL dataset. Pre-process the dataset as required. Compute Accuracy, Precision, Recall and F1 measure.

Text preprocessing

1. Load emails and labels (spam/ham or relevant classes). 2. Clean text: lowercase, remove punctuation, remove stopwords (optional), optionally apply stemming/lemmatization. 3. Convert to numeric features using TF-IDF (TfidfVectorizer) or CountVectorizer. 4. Split into train and test sets (e.g., 80/20).

Decision Tree training & metrics

Train sklearn.tree.DecisionTreeClassifier. After prediction compute accuracy, precision, recall, f1-score using sklearn.metrics. Use stratified split if classes are imbalanced. Consider grid search for max_depth/min_samples_split to avoid overfitting and use cross-validation.

Sample Python code

```
# Decision Tree on EMAIL dataset (example)
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classificat

df = pd.read_csv('emails.csv')  # columns: 'text','label'
X_text = df['text']
y = df['label']

tfidf = TfidfVectorizer(max_df=0.95, min_df=2, ngram_range=(1,2))
X = tfidf.fit_transform(X_text)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_stat

dt = DecisionTreeClassifier(random_state=42)
params = {'max_depth':[10,20,30,None], 'min_samples_split':[2,5,10]}
grid = GridSearchCV(dt, params, cv=5, scoring='f1_weighted', n_jobs=-1)
grid.fit(X_train, y_train)

best = grid.best_estimator_
y_pred = best.predict(X_test)

print('Accuracy:', accuracy_score(y_test,y_pred))
print('Precision (macro):', precision_score(y_test,y_pred, average='macro'))
print('Recall (macro):', recall_score(y_test,y_pred, average='macro'))
print('F1 (macro):', f1_score(y_test,y_pred, average='macro'))
print('\nFull classification report:\n', classification_report(y_test,y_pred))
```

Notes on evaluation

If dataset is imbalanced, report precision/recall per class and use weighted averages. Show confusion matrix. Discuss overfitting: deep trees may have high training accuracy but poor generalization.