Q3 - k-NN Classifier (EMAIL Dataset) - Detailed Answers

Problem statement

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

Preprocessing specifics for k-NN

k-NN uses distance — ensure features are scaled. For text data using TF-IDF, the vectors are already appropriate but you may want to use dimensionality reduction (TruncatedSVD) because k-NN on high-dimensional sparse data is expensive. Use cosine similarity (via metric='cosine') or Euclidean on dense embeddings.

Sample Python code

```
# k-NN on EMAIL dataset (example)
from sklearn.neighbors import KNeighborsClassifier
from sklearn.decomposition import TruncatedSVD
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import classification_report

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

pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(max_df=0.95, min_df=2, ngram_range=(1,2))),
    ('svd', TruncatedSVD(n_components=100, random_state=42)),  # reduce dims for kNN
    ('knn', KNeighborsClassifier())
])

params = {
    'knn__n_neighbors': [3,5,7,9],
    'knn__weights': ['uniform','distance'],
    'knn__metric': ['euclidean']
}

grid = GridSearchCV(pipeline, params, cv=5, scoring='f1_weighted', n_jobs=-1)
grid.fit(X_text, y)
print('Best params:', grid.best_params_)

X_train, X_test, y_train, y_test = train_test_split(X_text, y, test_size=0.2, stratify=y, random
y_pred = grid.predict(X_test)
print(classification_report(y_test, y_pred))
```

Practical tips

k-NN can be slow on large data. Consider approximate nearest neighbors (Faiss, Annoy) for scalability. Report metrics for several k and plot performance vs k.