In [160…
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.multiclass import OneVsRestClassifier
import pandas as pd
from sklearn.model_selection import KFold
from sklearn.metrics import precision_recall_fscore_support
import numpy as np
from iterstrat.ml_stratifiers import MultilabelStratifiedKFold
```

In [ ]:
```
# making dataset ready for ML

path = "/Local/Coding/project/data_clean/Full_dataset_coded.xlsx"
df = pd.read_excel(path)

label_cols = [
    "behavioural_advice_to_girls",
    "increase_information_awareness",
    "improve_service_access_or_quality",
    "support_girls_and_young_mothers",
    "community_engagement",
    "contraception_access_use_knowledge",
    "engaging_partners_in_srhr",
    "material_or_financial_support",
    "government_or_policy_action",
    "parental_engagement_and_responsibility",
    "violence_or_GBV_response",
    "no_recommendation",
    "unspecified_positive_recommendation"
]

df["n_labels"] = df[label_cols].sum(axis=1)
df["n_labels"].value_counts().sort_index()
```

Out[ ]:
```
n_labels
0    613
1    271
2     49
3     12
4      1
Name: count, dtype: int64
```

In [162…
```
df_labeled = df[df["n_labels"] > 0].copy()
len(df_labeled)
```

Out[162…
```
333
```

descriptive

1. Define text length and very_short

In [163…
```
df_ml = df_labeled.copy()
```

```
df_ml.shape
```

Out[163… (333, 24)

2. Dataset size (recommendations)

In [164…
```
n_total = len(df_ml)
n_usable = len(df_ml[~df_ml["very_short"]])

print(f"Total recommendation responses: {n_total}")
print(f"Usable (non-very short) responses: {n_usable}")
```

```
Total recommendation responses: 333
Usable (non-very short) responses: 333
```

3. Share of very short responses

In [165…
```
df_ml["very_short"].mean()
```

Out[165… np.float64(0.0)

4. Text length distribution (usable only)

In [166…
```
# create text_length if missing
if "text_length" not in df_ml.columns:
    df_ml["text_length"] = df_ml["text"].fillna("").astype(str).str

# now this works
df_ml.loc[~df_ml["very_short"], "text_length"].median()
```

Out[166… np.float64(57.0)

5. Category distribution (human-coded subset)

In [167…
```
label_cols = [
    "behavioural_advice_to_girls",
    "increase_information_awareness",
    "improve_service_access_or_quality",
    "support_girls_and_young_mothers",
    "community_engagement",
    "contraception_access_use_knowledge",
    "engaging_partners_in_srhr",
    "material_or_financial_support",
    "government_or_policy_action",
    "parental_engagement_and_responsibility",
    "violence_or_GBV_response",
    "no_recommendation",
    "unspecified_positive_recommendation"
]
label_counts = df_ml[label_cols].sum().sort_values(ascending=False)
label_counts
```

Out[167…
```
increase_information_awareness             92
improve_service_access_or_quality          70
support_girls_and_young_mothers            52
community_engagement                       35
unspecified_positive_recommendation        35
behavioural_advice_to_girls                34
no_recommendation                          26
parental_engagement_and_responsibility     14
violence_or_GBV_response                   14
contraception_access_use_knowledge         13
engaging_partners_in_srhr                   9
material_or_financial_support               9
government_or_policy_action                 6
dtype: int64
```

In [168…
```
# percentages:
(label_counts / len(df_ml) * 100).round(1)
```

Out[168…
```
increase_information_awareness             27.6
improve_service_access_or_quality          21.0
support_girls_and_young_mothers            15.6
community_engagement                       10.5
unspecified_positive_recommendation        10.5
behavioural_advice_to_girls                10.2
no_recommendation                           7.8
parental_engagement_and_responsibility      4.2
violence_or_GBV_response                    4.2
contraception_access_use_knowledge          3.9
engaging_partners_in_srhr                   2.7
material_or_financial_support               2.7
government_or_policy_action                 1.8
dtype: float64
```

6. Multi-label intensity (optional)

In [169…
```
df_ml["n_labels"] = df_ml[label_cols].sum(axis=1)
df_ml["n_labels"].value_counts().sort_index()
```

Out[169…
```
n_labels
1    271
2     49
3     12
4      1
Name: count, dtype: int64
```

In [ ]:
```
############################################################
# 2) Build ML dataframe and (optional) save it
############################################################
ml_df = df_ml[["id", "text"] + label_cols].copy()

out_path = "/Local/Coding/project/data_clean/recommendations_ml_rea
ml_df.to_excel(out_path, index=False)
print("Saved ML-ready dataset to:", out_path)

# IMPORTANT: build both X and Y from the SAME dataframe (ml_df)
```

```
X_text = ml_df["text"]
Y = ml_df[label_cols].astype(int).values
```

```
Saved ML-ready dataset to: /Users/anamargarida/Local/Coding/project/
data_clean/recommendations_ml_ready.xlsx
```

In [173…

```
############################################################
# 3) TF-IDF vectorization
############################################################
vectorizer = TfidfVectorizer(
    lowercase=True,
    stop_words="english",
    ngram_range=(1, 2),
    min_df=2,
    max_df=0.9,
    max_features=3000
)

X = vectorizer.fit_transform(X_text)
print("TF-IDF matrix shape:", X.shape)
```

```
TF-IDF matrix shape: (333, 381)
```

In [174…

```
############################################################
# Helper: run CV and return per-label averages
############################################################
def run_cv(splitter, X, Y, label_cols):
    results = {label: {"precision": [], "recall": [], "f1": []}
               for label in label_cols}

    for train_idx, test_idx in splitter.split(X, Y) if hasattr(spli
        X_train, X_test = X[train_idx], X[test_idx]
        Y_train, Y_test = Y[train_idx], Y[test_idx]

        clf.fit(X_train, Y_train)
        Y_pred = clf.predict(X_test)

        for i, label in enumerate(label_cols):
            p, r, f, _ = precision_recall_fscore_support(
                Y_test[:, i],
                Y_pred[:, i],
                average="binary",
                zero_division=0
            )
            results[label]["precision"].append(p)
            results[label]["recall"].append(r)
            results[label]["f1"].append(f)

    summary = []
    for label in label_cols:
        summary.append({
            "label": label,
            "precision": float(np.mean(results[label]["precision"])
            "recall": float(np.mean(results[label]["recall"])),
            "f1": float(np.mean(results[label]["f1"])),
        })
```

```
    return pd.DataFrame(summary).sort_values("f1", ascending=False)
```

In [175…

```
############################################################
# 5) Baseline A: standard KFold
############################################################
kf = KFold(n_splits=5, shuffle=True, random_state=42)
results_df_kfold = run_cv(kf, X, Y, label_cols)
print("\nBaseline A (KFold):")
display(results_df_kfold)
```

```
Baseline A (KFold):
```

	label	precision	recall	f1
11	no_recommendation	0.745000	0.858333	0.794913
2	improve_service_access_or_quality	0.709952	0.708936	0.699803
1	increase_information_awareness	0.667367	0.692460	0.673240
0	behavioural_advice_to_girls	0.590909	0.696825	0.613627
6	engaging_partners_in_srhr	0.533333	0.533333	0.520000
9	parental_engagement_and_responsibility	0.653333	0.470000	0.514286
12	unspecified_positive_recommendation	0.405000	0.558333	0.454286
10	violence_or_GBV_response	0.485714	0.566667	0.446234
3	support_girls_and_young_mothers	0.402880	0.526667	0.445104
4	community_engagement	0.431587	0.365714	0.382107
5	contraception_access_use_knowledge	0.336667	0.353333	0.333333
8	government_or_policy_action	0.300000	0.400000	0.333333
7	material_or_financial_support	0.280000	0.400000	0.314286

In [176…

```
############################################################
# 6) Baseline B: Multilabel stratified CV
############################################################
mskf = MultilabelStratifiedKFold(n_splits=5, shuffle=True, random_s
results_df_strat = run_cv(mskf, X, Y, label_cols)
print("\nBaseline B (MultilabelStratifiedKFold):")
display(results_df_strat)
```

```
Baseline B (MultilabelStratifiedKFold):
```

	label	precision	recall	f1
11	no_recommendation	0.827857	0.893333	0.843745
2	improve_service_access_or_quality	0.718959	0.671429	0.689432
1	increase_information_awareness	0.643667	0.650877	0.641394
6	engaging_partners_in_srhr	0.700000	0.700000	0.640000
0	behavioural_advice_to_girls	0.561111	0.704762	0.616197
8	government_or_policy_action	0.600000	0.500000	0.533333
9	parental_engagement_and_responsibility	0.673333	0.433333	0.496667
7	material_or_financial_support	0.500000	0.400000	0.433333
12	unspecified_positive_recommendation	0.387734	0.457143	0.414133
3	support_girls_and_young_mothers	0.378871	0.461818	0.411681
10	violence_or_GBV_response	0.400000	0.366667	0.380000
5	contraception_access_use_knowledge	0.316667	0.466667	0.375238
4	community_engagement	0.369048	0.314286	0.334852

In [ ]:
```
############################################################
# 7) Optional: save results tables
############################################################
out_a = "/Local/Coding/project/data_clean/results_baseline_A_kfold.
out_b = "/Local/Coding/project/data_clean/results_baseline_B_strati

results_df_kfold.to_excel(out_a, index=False)
results_df_strat.to_excel(out_b, index=False)

print("Saved Baseline A results to:", out_a)
print("Saved Baseline B results to:", out_b)
```