For better readability, view this project on the GitHub page.

This is a quick project that use machine learning for breast cancer recurrence prediction. The dataset is publicly available.

Table of Contents

- Overview
- Prerequisite
- Methods & Results
  - Data exploration & Summary table
  - Data pre-processing
  - Machine learning model training
  - Evaluation

Overview

We processed the raw data (N=286) and split it into a training set (80%) and a test set (20%). Four classic machine learning models were evaluated:

Model	Precision	Recall	F1	Accuracy	AUROC
Logistic Regression	0.3750	0.1875	0.2500	0.6842	0.6311
Ridge Regression	0.3750	0.1875	0.2500	0.6842	0.6372
Random Forest	0.3000	0.1875	0.2308	0.6491	0.5991
XGBoost	0.4167	0.3125	0.3571	0.6842	0.6601

Among them, the XGBoost shows best performance (precision, recall, F1 and AUROC).

Prerequisite

The required packages are listed in the requirements.txt. The important packages are - pandas - numpy - sklearn - xgboost - matplotlib

Methods & Results

Our method includes several steps: data exploration, data curation, summary table, machine learning model training and evaluation.

Data exploration & Summary table

Data exploration is done in the Jupyter Notebook explore_data.ipynb. Visual inspection were done to understand the data structure. Data types, missingness, range and distributes were accessed.

A summary table is created with the *tableone* package.

We notice that cancer recurrent: non-recurrent is about 1:3. There are predictors (e.g., inv-nodes, node-caps, deg-malig, irradiat) show significant correlation with the recurrence.

Grouped by class

Missing

Overall

no-recurrence-events

recurrence-events

P-Value

n

286

201

85

age, n (%)

20-29

1 (0.3)

1 (0.5)

0.550

30-39

36 (12.6)

21 (10.4)

15 (17.6)

40-49

90 (31.5)

63 (31.3)

27 (31.8)

50-59

96 (33.6)

71 (35.3)

25 (29.4)

60-69

57 (19.9)

40 (19.9)

17 (20.0)

70-79

6 (2.1)

5 (2.5)

1 (1.2)

menopause, n (%)

ge40

129 (45.1)

94 (46.8)

35 (41.2)

0.673

lt40

7 (2.4)

5 (2.5)

2 (2.4)

premeno

150 (52.4)

102 (50.7)

48 (56.5)

tumor-size, n (%)

0-4

8 (2.8)

7 (3.5)

1 (1.2)

0.056

10-14

28 (9.8)

27 (13.4)

1 (1.2)

15-19

30 (10.5)

23 (11.4)

7 (8.2)

20-24

50 (17.5)

34 (16.9)

16 (18.8)

25-29

54 (18.9)

36 (17.9)

18 (21.2)

30-34

60 (21.0)

35 (17.4)

25 (29.4)

35-39

19 (6.6)

12 (6.0)

7 (8.2)

40-44

22 (7.7)

16 (8.0)

6 (7.1)

45-49

3 (1.0)

2 (1.0)

1 (1.2)

5-9

4 (1.4)

4 (2.0)

50-54

8 (2.8)

5 (2.5)

3 (3.5)

inv-nodes, n (%)

0-2

213 (74.5)

167 (83.1)

46 (54.1)

<0.001

12-14

3 (1.0)

1 (0.5)

2 (2.4)

15-17

6 (2.1)

3 (1.5)

3 (3.5)

3-5

36 (12.6)

19 (9.5)

17 (20.0)

6-8

17 (5.9)

7 (3.5)

10 (11.8)

9-11

10 (3.5)

4 (2.0)

6 (7.1)

24-26

1 (0.3)

1 (1.2)

node-caps, n (%)

?

8 (2.8)

5 (2.5)

3 (3.5)

<0.001

no

222 (77.6)

171 (85.1)

51 (60.0)

yes

56 (19.6)

25 (12.4)

31 (36.5)

deg-malig, n (%)

1

71 (24.8)

59 (29.4)

12 (14.1)

<0.001

2

130 (45.5)

102 (50.7)

28 (32.9)

3

85 (29.7)

40 (19.9)

45 (52.9)

breast, n (%)

left

152 (53.1)

103 (51.2)

49 (57.6)

0.389

right

134 (46.9)

98 (48.8)

36 (42.4)

breast-quad, n (%)

central

21 (7.3)

17 (8.5)

4 (4.7)

0.319

left_low

110 (38.5)

75 (37.3)

35 (41.2)

left_up

97 (33.9)

71 (35.3)

26 (30.6)

right_low

24 (8.4)

18 (9.0)

6 (7.1)

right_up

33 (11.5)

20 (10.0)

13 (15.3)

?

1 (0.3)

1 (1.2)

irradiat, n (%)

no

218 (76.2)

164 (81.6)

54 (63.5)

0.002

yes

68 (23.8)

37 (18.4)

31 (36.5)

Data pre-processing

Based on the observations and understanding from data exploration, we pre-processed the data with this Python script.

Converting ordinal data to numeric Given the small sample size, we decide to represent the "range" data by the mid-point. This will reduce the dimention compared to encoding as one-hot vectors.

```
# ordinal to numeric
curated_df['age'] = df['age'].map({t:(int(t.split('-')[0]) + int(t.split('-')[1]))/2 for t i
curated_df['tumor-size'] = df['tumor-size'].map({t:(int(t.split('-')[0]) + int(t.split('-')
curated_df['inv-nodes'] = df['inv-nodes'].map({t:(int(t.split('-')[0]) + int(t.split('-')[1
curated_df['deg-malig'] = df['deg-malig'].astype(float)
```

Converting categorical data to one-hot We convert the true categorical data into one-hot encoding, while dropping the first category to avoid multi-collinearity. Note that dropping first category is optional for most machine learning models, but will help linear models' interpretability.

```
cate = pd.get_dummies(df[['menopause', 'node-caps', 'breast', 'breast-quad', 'irradiat']], 
```

Split data into training set and test set We randomly sample 20% of instances into a test set. This is not the best practise for predictive models since temporal effect is not considered. It would be nice to use more recent data as test set. In this case, since dates are not provided, we did random sampling.

```
""" Train-test split """
np.random.seed(123)
test_ids = np.random.choice(curated_df['id'], size=int(curated_df.shape[0] * 0.2), replace=
curated_df['train_test'] = (curated_df['id'].isin(test_ids)).map({True: 'test', False: 'tra
curated_df['train_test'].value_counts()
```

The final dataset

```
class                        bool
age                       float64
tumor-size                float64
inv-nodes                 float64
deg-malig                 float64
menopause_lt40               bool
menopause_premeno            bool
node-caps_no                 bool
node-caps_yes                bool
breast_right                 bool
breast-quad_central          bool
breast-quad_left_low         bool
breast-quad_left_up          bool
breast-quad_right_low        bool
breast-quad_right_up         bool
irradiat_yes                 bool
```

Machine learning model training

The training and evaluation pipeline is a Python script that runs in cmd. It takes a parameter `--config` or `-c` for each run. The configs for our experiments are available in this folder. For example:

```
python Train_eval_pipeline.py -c ./configs/RandomForestClassifier.yaml
```

When the pipeline runs, the training set and test set are loaded

```
feature_cols = ['age', 'tumor-size', 'inv-nodes', 'deg-malig',
    'menopause_lt40', 'menopause_premeno', 'node-caps_no', 'node-caps_yes',
    'breast_right', 'breast-quad_central', 'breast-quad_left_low',
    'breast-quad_left_up', 'breast-quad_right_low', 'breast-quad_right_up',
    'irradiat_yes']

train_features = df.loc[df['train_test'].isin(['train']), feature_cols]
train_labels = df.loc[df['train_test'].isin(['train']), 'class']

test_features = df.loc[df['train_test'].isin(['test']), feature_cols]
test_labels = df.loc[df['train_test'].isin(['test']), 'class']
```

Scaling is performed to boost model performance.

```
scaler = StandardScaler()
train_features_scaled = pd.DataFrame(scaler.fit_transform(train_features), columns=train_fea
test_features_scaled = pd.DataFrame(scaler.transform(test_features), columns=test_features.c
```

We train models with Sci-kit learn.

```
if config['model'] == "LogisticRegression":
        model = LogisticRegression(multi_class='multinomial', max_iter=500, penalty=None)
elif config['model'] == "RidgeRegression":
    model = LogisticRegression(multi_class='multinomial', max_iter=500, penalty='l2')
elif config['model'] == "RandomForestClassifier":
    model = RandomForestClassifier(n_estimators=100, max_depth=None, random_state=123)
elif config['model'] == "XGBClassifier":
    model = XGBClassifier(random_state=123)
else:
    raise ValueError(f"model {config['model']} is not supported.")

model.fit(train_features_scaled, train_labels)
```

Evaluation

The evaluation metrics include precision, recall, F1, accuracy, and AUROC.

```
y_prob = model.predict_proba(test_features_scaled)
y_pred = model.predict(test_features_scaled)
gold = test_labels

metrics = []
precision = precision_score(gold, y_pred)
recall = recall_score(gold, y_pred)
f1 = f1_score(gold, y_pred)
acc = accuracy_score(gold, y_pred)
auroc = roc_auc_score(gold, y_prob[:,1])
metrics.append({"Precision": precision,
                "Recall": recall,
                "F1": f1,
                "Accuracy": acc,
                "AUROC": auroc})
metrics_df = pd.DataFrame(metrics)
```

We also plot an overall ROC curve with this script.