1: Main Work

(a) Process Data:

The first is to read the data and analyze:

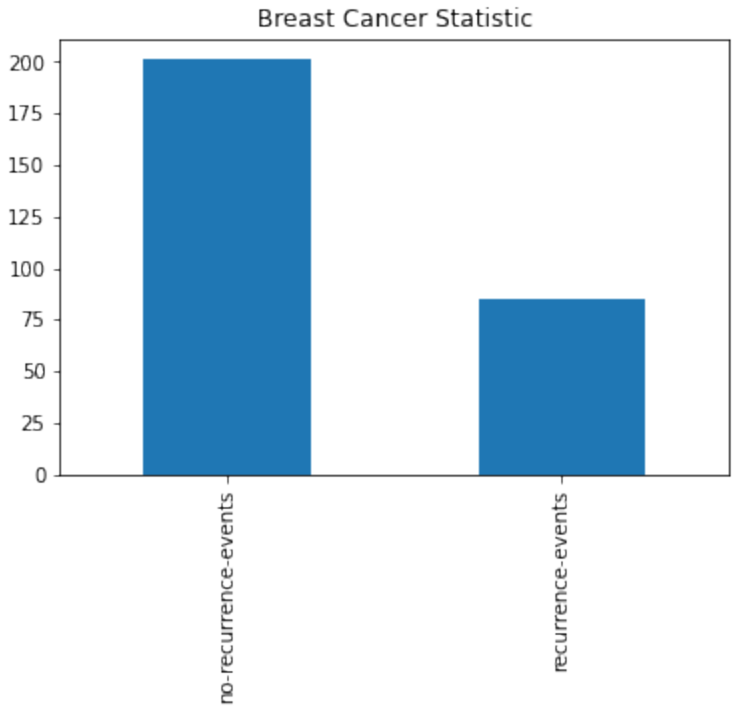


Figure 1: Breast Cancer Statistic

It is analyzed from Figure 1 that this is an unbalanced data set, which needs to be paid attention to when processing data.
Then analyze each group of feature data and find that some data are missing, and some features are not convenient for classification processing.

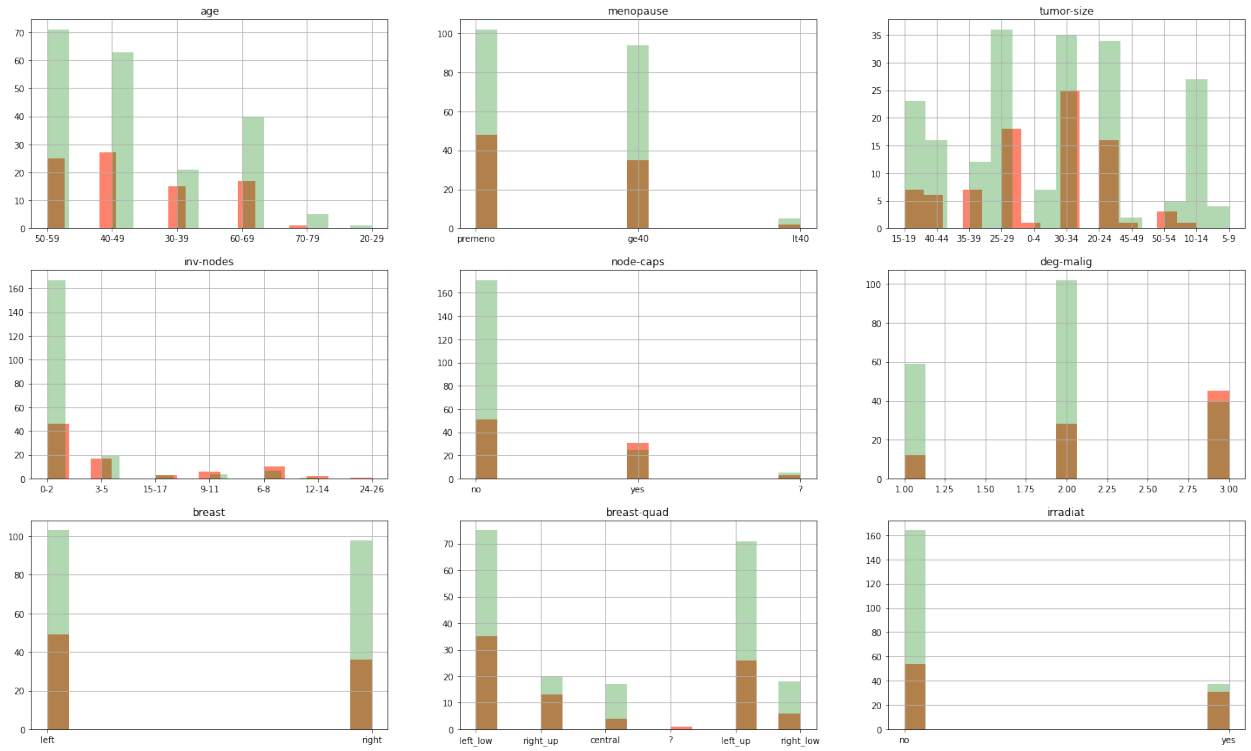
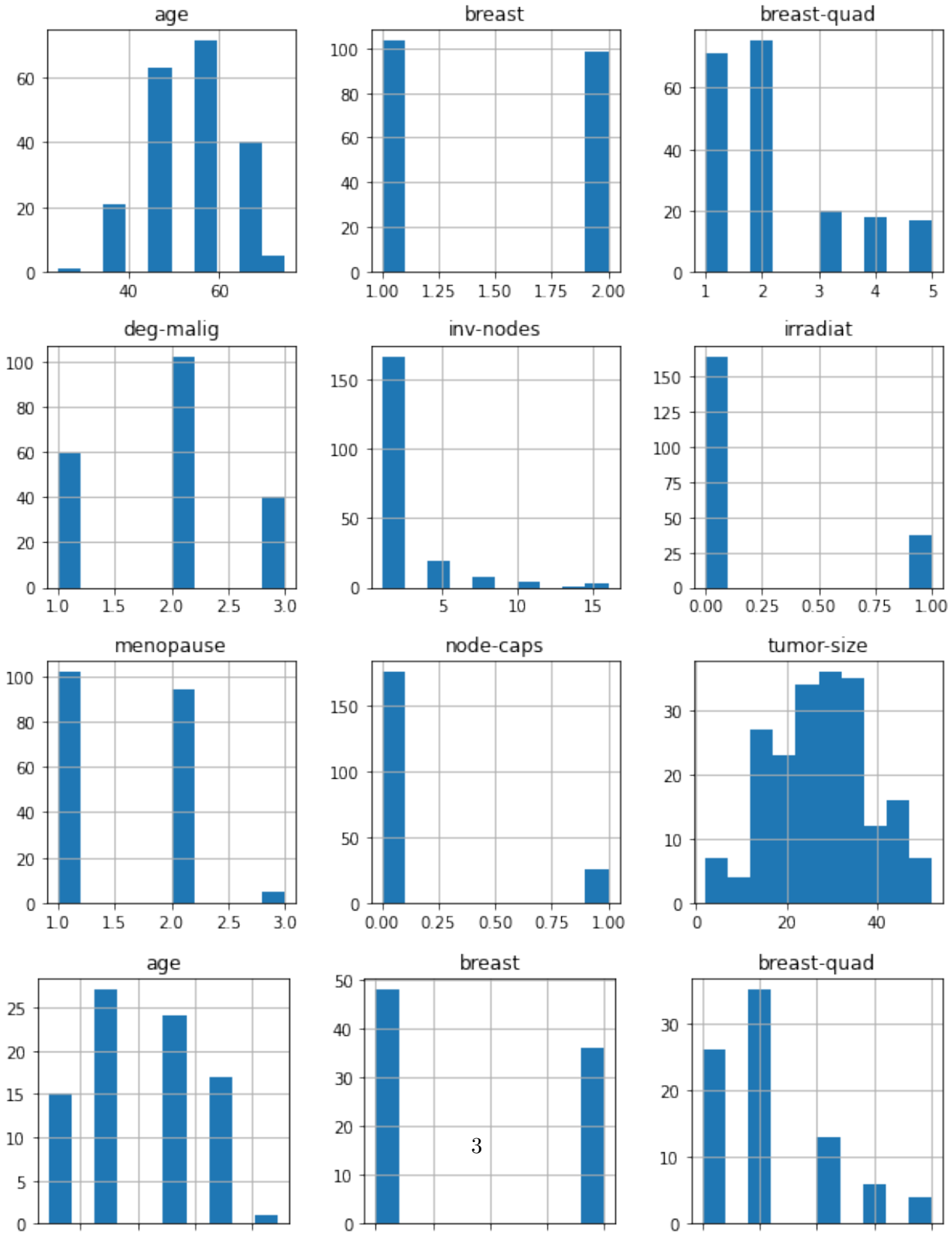


Figure 2: Visualize Data

Subsequent feature processing to facilitate classification, the main steps include but are not limited to binarize Class & node-caps & irradiat ; binarize Class & node-caps & irradiat; convert inv-nodes to the median of data; convert age to the numerical average of data; convert tumor-size to the numerical average of data. The results after processing are as follows:

	Class	age	menopause	tumor-size	inv-nodes	node-caps	deg-malig	breast	breast-quad	irradiat
0	0	34.5	1	32	1	0	3	1	2.0	0
1	0	44.5	1	22	1	0	2	2	3.0	0
2	0	44.5	1	22	1	0	2	1	2.0	0
3	0	64.5	2	17	1	0	2	2	1.0	0
4	0	44.5	1	2	1	0	2	2	4.0	0



Also use randomforeast to analyze top features:

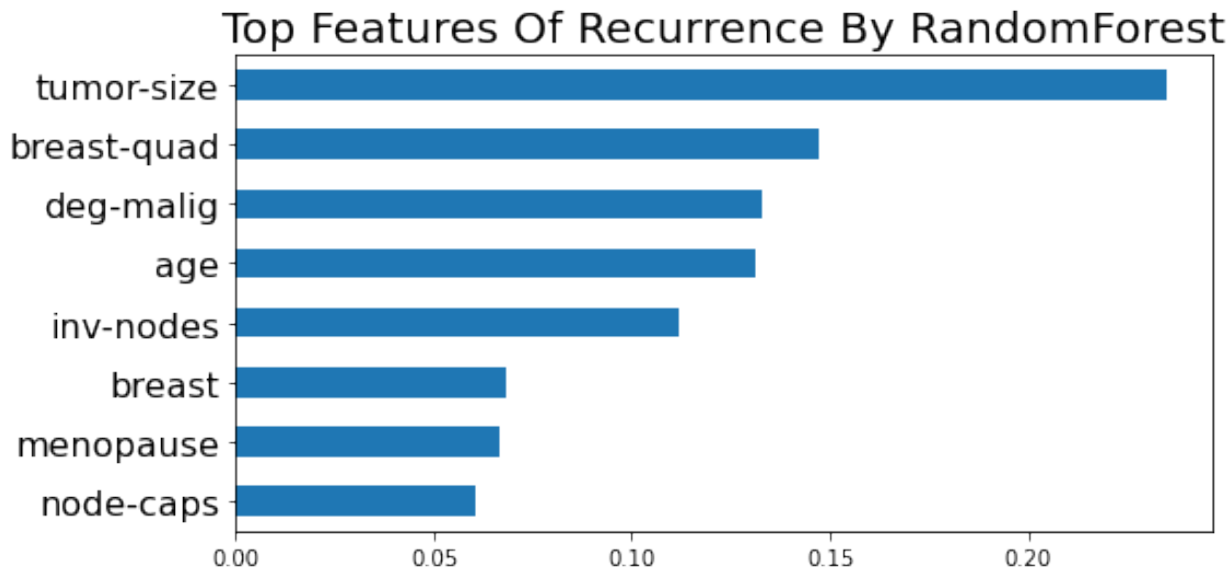


Figure 4: Top Features

At last, it can be analyzed from Figures 2, 3, and 4 that through filtering, 6 features relationships are retained — ['age','menopause','tumor-size','inv-nodes','node-caps','deg-malig'].

(b) Perform Model Selectio and Optimize:

First split data to 70:30 ratio, and use StratifiedKFold for imbalance class. I selected three models of LogisticRegression, DecisionTree, GaussianNB to train, and evaluated from 6 dimensions such as recall rate.

	model	acc	recall	f1	rocauc	logloss	timetaken
0	Logistic	0.728462	0.322727	0.413968	0.722646	11.245248	0
1	DecisionTree	0.648205	0.404545	0.398841	0.572835	11.245304	0
2	GaussianNB	0.733718	0.442424	0.498656	0.726109	10.442039	0
3	KNN	0.678205	0.216667	0.273220	0.624811	12.450108	0
4	RandomForest	0.658462	0.404545	0.411924	0.669183	10.843672	0
5	SVC	0.703590	0.000000	0.000000	0.697051	10.040342	0

Figure 5: Output

And then optimise model: hyperparameter tuning

```
=====DecisionTree=====
optimal F1 score = 0.4189
optimal max_depth = 2
optimal F1 score = 0.5000
optimal threshold = 0.080
DecisionTree acc score is
Training: 77.89%
Test set: 76.90%

Adjust to 0.25:
Precision: 0.4800,   Recall: 0.4800,   F1 Score: 0.4800
DecisionTree confusion matrix:
 [[48 13]
 [13 12]]

Default 0.50:
Precision: 0.5714,   Recall: 0.1600,   F1 Score: 0.2500
DecisionTree confusion matrix:
 [[58  3]
 [21  4]]

Adjust to 0.75:
Precision: 0.5714,   Recall: 0.1600,   F1 Score: 0.2500
DecisionTree confusion matrix:
 [[58  3]
 [21  4]]

Optimal threshold 0.080
Precision: 0.3433,   Recall: 0.9200,   F1 Score: 0.5000
DecisionTree confusion matrix:
 [[17 44]
 [ 2 23]]
DecisionTree Log-loss: 0.5740
```

	model	acc	recall	f1	rocauc	logloss	timetaken
0	Logistic	0.728462	0.80	0.540541	0.721943	0.571700	35
1	DecisionTree	0.768974	0.92	0.500000	0.666964	0.573984	0
2	GaussianNB	0.733718	0.72	0.571429	0.726109	0.939928	0
3	KNN	0.678590	0.76	0.413043	0.636661	2.934723	1
4	RandomForest	0.698590	0.60	0.508475	0.666044	0.673368	44
5	SVC	0.703590	0.72	0.507042	0.697051	0.586078	0

Figure 6: Optimized Output

(c) Compare Results: After the comparison and optimization, the three models have achieved better results than before, and each evaluation dimension has been improved accordingly. **At the same time, the best model should be DT**

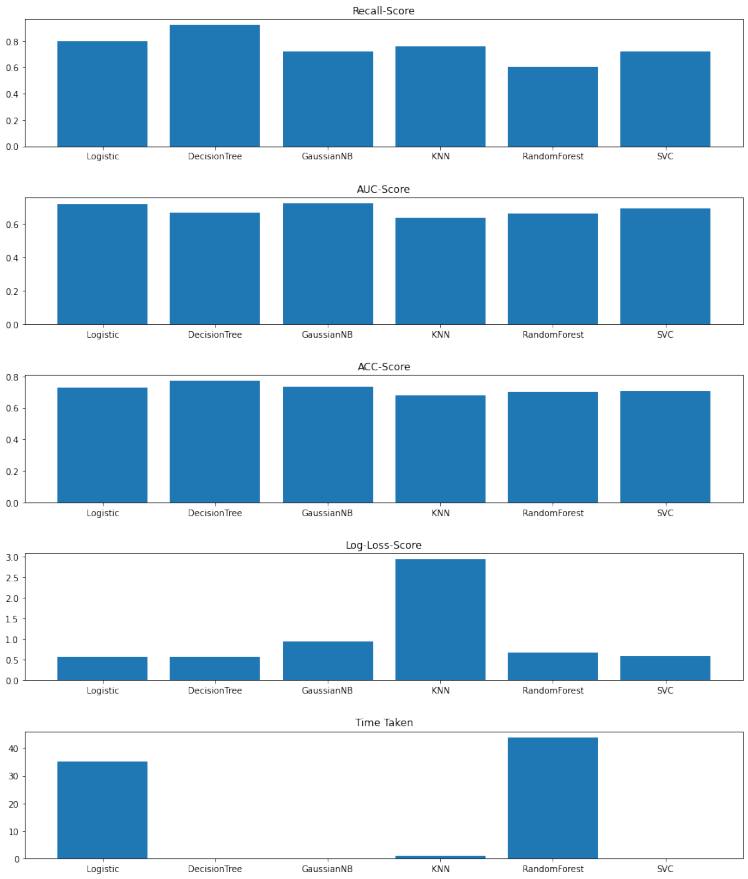


Figure 7: Evalute Result

2: Related Theory

Logistic Regression: The core of the regression method is to find the most suitable parameters for the function, so that the value of the function is closest to the value of the sample. For example, linear regression is to find the most suitable a and b for the function f (x) = ax + b.

LR does not fit a linear function. It fits a function in probability. The value of f (x) now reflects the probability that the sample belongs to this class.

Applicable scenarios:

LR is also the basic component of many classification algorithms. Its advantage is that the output value naturally falls between 0 and 1 and has a probabilistic significance.

Because it is essentially a linear classifier, it cannot handle the correlation between features.

Decision Tree: The characteristic of a decision tree is that it is always segmenting along features. As the layers progress, this division will become more and more fine.

Although the generated tree is not easy for the user to see, when analyzing the data, by observing the upper structure of the tree, you can have an intuitive experience of the core idea of the classifier.

Applicable scenarios:

Because it can generate a clear tree structure that selects different prediction results based on features, data analysts often use decision trees when they want to better understand the data at hand.

GaussianNB: In all machine learning classification algorithms, Naive Bayes is different from most other classification algorithms. For most classification algorithms, such as decision trees, KNN, logistic regression, support vector machines, etc., they are all discriminant methods, that is, directly learning the relationship between the feature output Y and the feature X, or the decision function Y = f (X), or the conditional distribution P (Y | X). However, Naive Bayes is a generation method, that is, to directly find the joint distribution P (X, Y) of the feature output Y and the feature X, and then use P(Y) inferred.

Applicable scenarios:

GaussianNB is generally used to process sample features, most of which are continuous values.

Based on the above foundation, the optimization method is searching for optimal threshold, vary from 0.0001 to 0.9999, fit/predict on train/test data.

1. Search for optimal hyperparameter C in LogisticRegresssion, vary C from 0.001 to 1000, using KFold(5) Cross Validation on train data.
2. Search for optimal max depth in DecisionTree, vary 2 to 10, using KFold(8) cross validation on train data

```
print('\n=====LogisticRegression=====')
time1 = time.time()
kf = KFold(n_splits=5, random_state=SEED, shuffle=True)
score_list = []
c_list = 10**np.linspace(-3,3,300)
for c in c_list:
    logit = LogisticRegression(C = c)
    cvs = (cross_val_score(logit,x_train, y_train, cv=kf, scoring='f1')).mean()
    score_list.append(cvs)
print('optimal cv F1 score = {:.4f}'.format(max(score_list)))
optimal_c = float(c_list[score_list.index(max(score_list))])
print('optimal value of C = {:.3f}'.format(optimal_c))

logic = LogisticRegression(C = optimal_c)
model1 = optimise_record(logic, x_train, x_test, y_train, y_test, 'Logistic')
model1.timetaken[0] = time.time() - time1

print('\n=====DecisionTree=====')
time1 = time.time()
kf = KFold(n_splits=8, random_state=SEED, shuffle=True)
d_scores = []
for d in range(2, 11):
    decisiontree = DecisionTreeClassifier(max_depth=d)
```

```
    cvs = cross_val_score(decisiontree, x_train, y_train, cv=kf, scoring='f1').mean
    d_scores.append(cvs)
print('optimal F1 score = {:.4f}'.format(max(d_scores)))
optimal_d = d_scores.index(max(d_scores))+2
print('optimal max_depth =', optimal_d)

decisiontree = DecisionTreeClassifier(max_depth=optimal_d)
model2 = optimise_record(decisiontree, x_train, x_test, y_train, y_test, 'DecisionT
model2.timetaken[0] = time.time() - time1

print('\n=====GaussianNB=====')
time1 = time.time()
gnb = GaussianNB()
model3 = optimise_record(gnb, x_train, x_test, y_train, y_test, 'GaussianNB')
model3.timetaken[0] = time.time() - time1
```

3: Some Comments

The data of this project is relatively easy to handle. It should be noted that the data should also be normalized when processing the data. For the data of the range category, it is better to use the median to replace it for further classification.

Because the features are clear, DT may achieve better results.In fact, each model has achieved good results, but considering that this is an extremely imbalance data set, we should focus on the recall rate rather than the accuracy rate, so the best model is DecisionTree, and DT is still achieved gratifying resultson other data. At the same time, DT has achieved good results in acc, auc, loss, and timetaken.

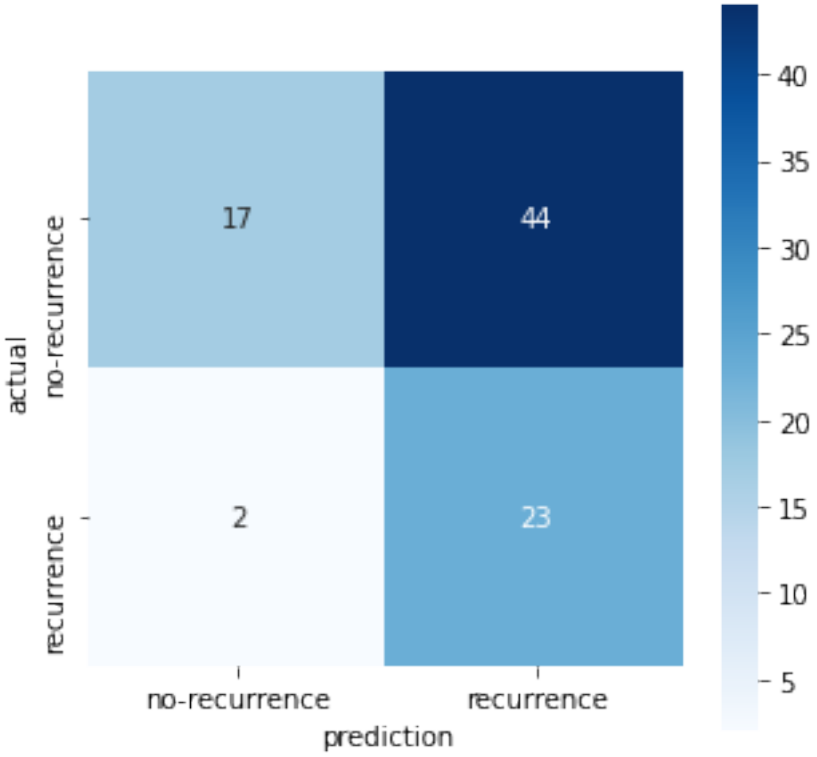


Figure 8: DT Confusion Matrix