Big Data Science Homework 5 - Yuxiang Huang

1 Preprocessing

1.1 Encoding

Encode those text features.

T Stage	N Stage	differentiate
T1	N1	Poorly
T2	N2	Moderately
T3	N3	Moderately

T Stage	N Stage	differentiate
0	0	1
1	1	2
2	2	2

Specially, use One Hot Encoding on features "Race" and "Marital Status" since these labels have no specific order of preference.

1.2 Outliers and Missing values

By looking through, there isn't obvious outlier in the dataset and it does not contain missing value either.

1.3 Normalization

Do Z-score normalization. For each value x in the dataset, replace it with $x' = \frac{x-\overline{x}}{\sigma}$

T Stage	N Stage	differentiate
0	0	1
1	1	2
2	2	2

T Stage	N Stage	differentiate
−1.03	−0.63	−1.33
0.28	0.81	0.24
1.59	2.25	0.24

1.4 Reduction

After encoding, there are 21 columns in the dataset (except for "Status", which we're going to predict). Use PCA module from sklearn library in Python, output the eigenvalues of the covariance matrix.

$\lambda 1$	$\lambda 2$	$\lambda 3$	$\lambda 4$	...	$\lambda 20$	$\lambda 21$
3.756	2.135	1.903	1.742	...	1.93e-30	1.77e-31

According to these eigenvalues, pick the first 12 principle components, concatenate them with "Status" column and get the dataset after dimensionality reduction. Now the dataset looks like:

PC1	PC2	...	PC12	Status
−1.123	0.618	...	−0.355	1
0.545	−1.151	...	−0.467	0

All the following algorithms (except for feature selection) make predection based on the reduced dataset.

2 Modeling

2.1 Feature Selection

Use Information Gain to complete feature selection by ranking features. Instead of doing feature selection on the reduced dataset, we do it on the original dataset since Information Gain requires discrete features while the features after reduction are continuous.

H(Status)	IG(Survival Months)	IG(Reginol Node Positive)	IG(6th Stage)
0.617	0.220	0.049	0.044

IG(N Stage)	IG(Tumor Size)	IG(Progesterone Status)	IG(differentiate)
0.041	0.033	0.019	0.019

IG(Grade)	IG(Estrogen Status)	IG(T Stage)	IG(Age)
0.019	0.018	0.017	0.014

IG(Regional Node Examined)	IG(A Stage)	IG(Race)	IG(Marital Status)
0.012	0.005	0.004	0.004

2.2 Modeling

Use split validation, divide the reduced dataset into a 70% training set and a 30% testing set.

Model	Accuracy on training set	Accuracy on testing set
KNN	0.900	0.874
Naïve Bayes	0.845	0.862
Decision Tree	1.0	0.831
Random Forest	0.999	0.896
Gradient Boosting	0.928	0.891

2.2.1 KNN (accuracy: 87.4%)

- Hyperparameters: K = 5
- Pros: 1) No Training Period. 2) One hyper paramete.
- Cons: 1) Computationally expensive during prediction, especially with large datasets. 2) Sensitive to irrelevant features and noisy data. 3) Requires storage of all training data

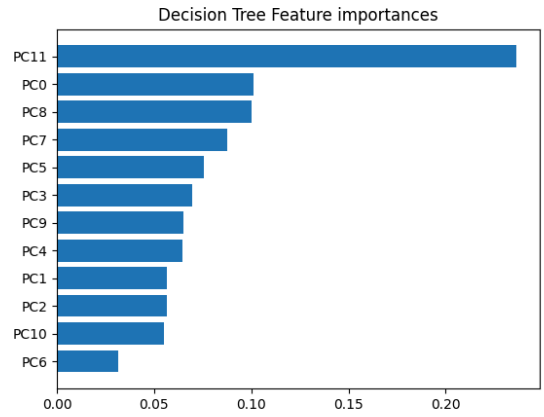
2.2.2 Naive Bayes (accuracy: 86.4%)

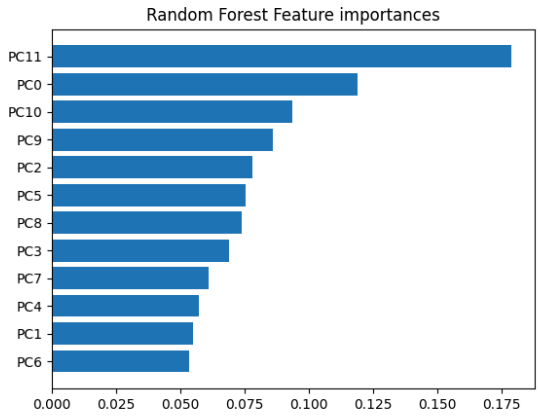
- Hyperparameters: var_smoothingfloat = 1e-9
- Pros: 1) Performs well with high-dimensional data. 2) Efficient, especially with large datasets.
- Cons: 1) Assumes independence among features. 2) Zero probability problem.

2.2.3 Decision Tree (accuracy: 83.1%)

- Hyperparameters: max_depth = None, min_samples_split = 2, min_samples_leaf = 1, max_features = None, random_state = None, max_leaf_nodes = None
- Pros: 1) Can handle both numerical and categorical data. 2) Intuitive and easy to explain to technical teams as well as stakeholders.

- Cons: 1) Decision tree often involves higher time to train the model. 2) A small change in the data can cause a large change in the structure of the decision tree causing instability.



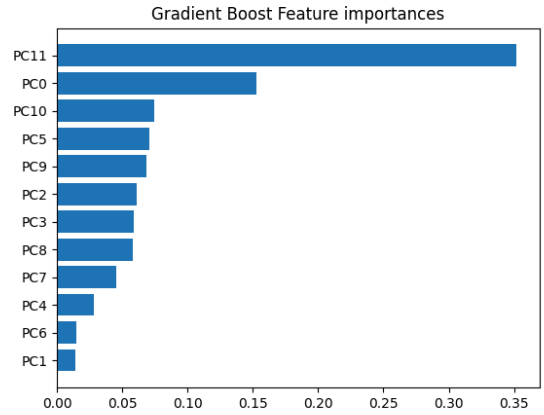


2.2.4 Random Forest (accuracy: 89.6%)

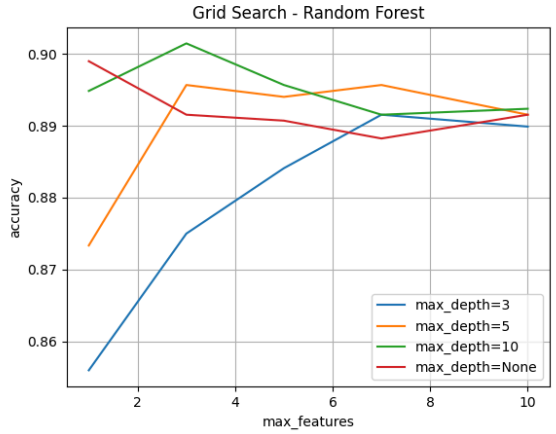
- Hyperparameters: max_depth = None, min_samples_split = 2, min_samples_leaf = 1, min_weight_fraction_leaf = 0, max_features = sqrt(n_features), max_leaf_nodes = None
- Pros: 1) By averaging or combining the outputs of various decision trees, random forests solve the overfitting issue. 2) Can be used for both classification and regression tasks. 3) Outliers do not significantly impact random forests.
- Cons: 1) Random forest construction is much more time- and labor-intensive than decision tree construction. 2) It takes a long time compared to other algorithms.

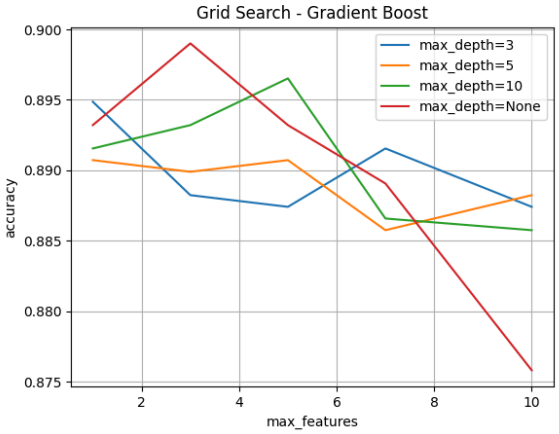
2.2.5 Gradient Boosting (accuracy: 89.1%)

- Hyperparameters: learning_rate = 0.1, n_estimators = 100, subsample = 1.0, min_samples_split = 2, min_samples_leaf = 1, max_depth = 3, validation_fraction = 0.1
- Pros: 1) Train faster especially on larger datasets. 2) Most of them provide support handling categorical features 3) Some of them handle missing values natively.
- Cons: 1) Prone to overfitting. 2) Limited scalability and less flexibility.



3 Hyperparameter Tuning





Pick Random Forest and Gradient Boosting to perform a hyperparameter search using Grid search.

- Max Depth: [3,5,10,None]
  The parameter "max_depth" represents the maximum level of each tree in the random forest model. A deeper tree performs well and captures a lot of information about the training data, but will not generalize well to test data.
- Max Features: [1,3,5,7,10]
  The parameter "max_features" represents the random forest model is allowed to try at each split. By default in Scikit-Learn, this value is set to the square root of the total number of variables in the dataset.

According to the figures above, for Random Forest algorithm, it achieves the best performance when max depth = 10 and max features = 3. For Gradient Boostring algorithm, it achieves the best performance when max depth = None and max features = 3

4 Conclusions

This report summarizes the hands-on experience in using feature selection and classification algorithms for the problem of breast cancer survivability prediction in SEER database.
As for preprocessing, I look through data for outliers and perform normalization. Also, I apply PCA reduction since the dataset has a lot of features.
As for modeling, I apply Information Gain which is a entropy based technique for feature ranking. Subsequently, I train 5 algorithms (KNN, Naive Bayes, Decision Tree, Random Forest, and Gradient Boost) on the dataset, which turns out that Random Forest achieves the best performance.
Finally, as for hyperparameter tuning, I use grid search to find the hyperparameters that perform the best.

Github Repository: https://github.com/Hyxpillow/BDS-spring2024-hw5