Predicting Breast Cancer Survivability Using Data Mining Techniques

This assignment focuses on application of data mining techniques to the Surveillance, Epidemiology, and End Results (SEER) Public-Use Data to predict the survivability rate of breast cancer patients. We will develop multiple predictive models to find one that can accurately classify patients into survivability outcomes based on a set of features derived from the SEER database. We will first preprocess the data, conduct feature engineering and selection. Then we will build models and use hyperparameter tuning to find the best hyperparameters. Finally, we will compare performances of different models to pinpoint which one is performing best for our task.

Feature Engineering and Preprocessing

Our first step was data exploration which included getting to know our features and looking for missing values or outliers.

	Age	Race	Marital Status	T Stage	N Stage	6th Stage	differentiate	Grade	A Stage	Tumor Size	Estrogen Status	Progesterone Status	Regional Node Examined	Reginol Node Positive	Survival Months	Status
0	68	White	Married	T1	N1	IIA	Poorly differentiated	3	Regional	4	Positive	Positive	24	1	60	Alive
1	50	White	Married	T2	N2	IIIA	Moderately differentiated	2	Regional	35	Positive	Positive	14	5	62	Alive
2	58	White	Divorced	T3	N3	IIIC	Moderately differentiated	2	Regional	63	Positive	Positive	14	7	75	Alive
3	58	White	Married	T1	N1	IIA	Poorly differentiated	3	Regional	18	Positive	Positive	2	1	84	Alive
4	47	White	Married	T2	N1	IIB	Poorly differentiated	3	Regional	41	Positive	Positive	3	1	50	Alive

From this table we can see that we are dealing with 16 individual features. All of the values in our dataset are present - there are no missing values. Furthermore, we have 4024 instances. We can notice that we have some numerical columns such as age, grade, tumor size, as well as some categorical variables such as race and marital status. Moreover, we also have some binary features such as estrogen and progesterone status as well as our target variable Status. Our target variable can either have a value of “Alive” or “Dead”.

After conducting a comprehensive inspection of our data our next step was transforming features so we are able to build predictive models. Specifically, we transformed 3 binary categorical columns to 1s and 0s and we used one-hot-encoding on the remaining categorical variables. After this we had 38 unique columns. After that, we decided to also conduct outlier removal as there were a wide range of variables and we wanted to make sure there have not been any outliers or human errors. Thus, we decided to remove top and bottom 2.5% of data values for numerical features. This left us with 3391 data instances.

Next, we normalized our numerical features to make sure all variables are on the same scale for the modeling stage as different scales can make our models less precise. For this we used a standard scaler.

Feature Selection and Ranking

As we are working with medical data and predicting survival outcomes we want to make sure our models are as interpretable as possible. Due to this we decided against dimensionality reduction methods like PCA or SVD. However, we will use a mix of Feature Selection and Ranking to reduce the number of features as 38 features is a bit excessive. Our first step here was to conduct Sequential Forward Selection with Cross-Validation over varying numbers of features to see at which point dropping more features significantly decreases our accuracy. The results show that after dropping more than 15 features our accuracy drops under 90% so we decided to retain the best 15 features according to Sequential Forward Selection. These were the features we retained due to Sequential Forward Selection: ['Survival Months', 'Race_Black', 'Race_Other', 'Race_White', 'Marital Status_Separated', 'T Stage _T4', 'N Stage_N3', '6th Stage_IIIB', '6th Stage_IIIC', 'differentiate_Moderately differentiated', 'differentiate_Undifferentiated', 'Grade_ anaplastic; Grade IV', 'Grade_2', 'A Stage_Distant', 'A Stage_Regional']

Similarly, we used SVM ranker to retain the best 15 features with highest rankings. These features included: ['Survival Months', 'differentiate_Undifferentiated', 'Grade_ anaplastic; Grade IV', 'T Stage _T4', 'Estrogen Status', 'Progesterone Status', 'T Stage _T1', 'differentiate_Well differentiated', 'Grade_1', 'Race_Black', 'N Stage_N1', 'Race_Other', 'differentiate_Moderately differentiated', 'Grade_2', '6th Stage_IIIA'].

After getting features from both methods we decided to retain their union - all the features that appeared in each of the methods. This has resulted in 22 relevant features (in addition to target variable).

At the end of feature selection we were left with 23 features to build models on (including the target feature).

Models and Results

Using the now processed dataset we built 5 models, KNN, Naïve Bayes, C4.5 Decision Tree, Random Forest, and Gradient Boosting.

KNN is a simple, instance-based learning algorithm that classifies a new sample based on the majority vote of its k nearest neighbors. The pros of KKN include the fact that it is quite simple and easy to implement and comprehend. Further, it does not make any assumptions about data distribution. However, as it does not “build” a model but compares each new instance to all training instances it can be very computationally intensive during testing, especially with large datasets. We can notice this by the fact it runs longer than all other models even though it is the simplest. Further, KNN can be very sensitive to irrelevant features and the scale of the data. The main hyperparameters include number of neighbors (k) that we are using for the majority vote and distance metric we are using, in our case Euclidean distance.

Naïve Bayes is a probabilistic classifier that applies Bayes' theorem with the “naive” assumption of conditional independence between every pair of features given the class. Its advantage is that it is fast and efficient, and it works well with high-dimensional data. Additionally, it performs well with categorical input variables. However, it assumes feature independence, which is rarely the case in real-world data so it might miss some relevant relationships. Thus, it can perform poorly with numerical features or data with strong feature correlations. The main hyperparameters include additive (Laplace/Lidstone) smoothing parameter (alpha), whether to learn class prior probabilities or not (fit_prior), portion of the largest variance of all features added to variances for stability (var_smoothing). In our case we used default sklearn parameters to build our model.

C4.5 is an extension of the basic Decision Tree algorithm that uses information gain or gain ratio to decide which feature to split on at each step in the tree. Its biggest advantage is that it generates human-readable rules and can handle both numerical and categorical data. However, it is prone to overfitting in cases of noisy or complex datasets and can be less accurate compared to other more complex models. This model is not directly available in sklearn but we adjusted hyperparameters in regular DecisionTreeClassifier by using “entropy” criterion to mimic it. Other main hyperparameters include the maximum depth of the tree (max_depth), minimum number of samples required to split an internal node (min_samples_split), and minimum number of samples required to be at a leaf node (min_samples_leaf).

Random Forest is an ensemble learning method that operates by constructing multiple decision trees during training and outputs the mode (most common) of the classes for classification tasks. It reduces overfitting through averaging multiple trees and can handle a large number of features and is robust to outliers. However, due to using many trees it can be slow in prediction and is much less interpretable compared to a single decision tree. Its main hyperparameters include number of trees in the forest (n_estimators), maximum number of features considered for splitting a node (max_features), and maximum depth of each decision tree (max_depth).

Gradient Boosting is a machine learning technique that produces a prediction model in the form of an ensemble of weak prediction models, most commonly decision trees. Its main advantage is that it provides the best accuracy that cannot be bettered and has a lot of flexibility. In other words, Gradient Boosting can optimize on different loss functions and provides several hyperparameter tuning options that make the function fit very flexible. However, due to such complexity it can overfit on datasets with noisy data and is very computationally expensive and takes longer to train. Its main hyperparameters include number of boosting stages to be run (n_estimators), learning rate that shrinks the contribution of each tree by this value (learning_rate), and maximum depth of each tree (max_depth).

The results from all of these models can be seen in this table. We will further discuss the results in the conclusion section.

Model Name	Accuracy	Precision	Recall	F1 Score
KNN	0.8954	0.7357	0.6881	0.7080
Naïve Bayes	0.8100	0.6024	0.6128	0.6070
C4.5 Decision Tree	0.8498	0.6748	0.6781	0.6764
Random Forest	0.8719	0.7182	0.6908	0.7030
Gradient Boosting	0.9131	0.8629	0.7193	0.7665

Hyperparameter Tuning Results

We used the Grid Search method for hyperparameter tuning of Random Forest and Gradient Boosting models. This method works by defining a "grid" of hyperparameters and then the grid search algorithm systematically constructs and evaluates one model for each combination of hyperparameter values in order to find the combination that performs the best according to a specified metric (such as accuracy). It is worth noting that grid search can be computationally expensive, especially when the hyperparameter space is large or the model takes a long time to train, because it needs to train and evaluate the model once for each combination. Despite this drawback, grid search is a commonly used method because of its simplicity and its ability to methodically work through multiple combinations.

We used grid search on a small space of hyperparameters for tuning of both Random Forest and Gradient Boosting models. Specifically, for the Random Forest model we created a grid search on 3 different parameters for number of estimators, 2 different parameters of maximum features and 3 different parameters of maximum tree depth. Similarly, for the Gradient Boosting model we created a grid search on 3 different parameters for number of estimators, 3 different learning rate parameters and 3 different parameters of maximum tree depth.

This table shows the results of above models as well as the results of random forest and gradient boosting models after hyperparameter tuning. We can notice that hyperparameter tuning led to slight increase in all the results for both Random Forest and Gradient Boosting. We can see that this increase was more significant for Random Forest compared to Gradient Boosting. However, since the increase was so slight for both models it is worth considering if the hyperparameter tuning is worth the time and resources for such a minimal increase in performance.

Model Name	Accuracy	Precision	Recall	F1 Score
KNN	0.8954	0.7357	0.6881	0.7080
Naïve Bayes	0.8100	0.6024	0.6128	0.6070
C4.5 Decision Tree	0.8498	0.6748	0.6781	0.6764
Random Forest	0.8719	0.7182	0.6908	0.7030
Gradient Boosting	0.9131	0.8629	0.7193	0.7665
Random Forest HT	0.9043	0.8272	0.7048	0.7460
Gradient Boosting HT	0.9146	0.8710	0.7201	0.7690

Conclusions

Our exploration into the prediction of breast cancer patient survivability using SEER Public-Use Data has included comprehensive preprocessing, feature selection, model development, and hyperparameter tuning and has resulted in a set of predictive models, each with its own merits and limitations. We can see from the above table that all the models have accuracy of over 80% with the best model before and after hyperparameter tuning being the Gradient Boosting model. This is not surprising as we mentioned earlier, Gradient Boosting models often provide unbeatable performance but it comes at the cost of interpretability and computing complexity (and potentially overfitting). We can see that this model performs only slightly better than Random Forest model (after hyperparameter tuning) and KNN (without tuning). This is interesting since KNN is the simplest model we can build and gives us the most interpretability out of them all. However, it is worth noting that (looking just at accuracy) KNN would be a fair choice because its accuracy is just about 2% worse than the best model, Tuned Gradient Boosting model, but it would have its own issues as it takes a long time to make predictions. This is because it does not "learn" from the data but just compares each new point with all the training ones which is costly, in fact even though it is the simplest model it ran for the longest time out of them all. Further, while the difference in accuracy between KNN, Random Forest and Gradient Boosting models is minimal, Random Forest and especially Gradient Boosting perform much better when it comes to Precision and Recall and with that also F1 score. However, using these models will cost us interpretability that KNN offers. We could potentially in the future look at SHAP values to see if we can get some of the interpretability back from more complex models.

Going back to feature selection and ranking, we saw that both methods highlighted Survival Months as the most relevant features however when it comes to the next best features Sequential Forward Selection focused more on demographic features (e.g. race and marriage status) while SVM focused more on patients medical information (e.g. grade of anaplastic, T stage, progesterone status). When looking at the features our models considered most relevant, we can see that all of them have considered 'Survival Months' as the most important feature followed by clinical factors including 'T Stage', 'N Stage', 'Progesterone Status' and only then followed by the 'Race' variable. This is good as it shows us that while patient demographic features do have influence on their diagnosis they are not as important as their clinical details. This emphasizes the importance of a multifaceted approach in understanding patient outcomes, where biological, clinical, and demographic features collectively shape predictive modeling.

At the end we have built 5 models with very good predictive performance and we have to decide whether to choose models with higher performance metrics or more interpretable based on our needs and preferences. I believe that KNN would be a good choice if time and computing needs were not an issue, but if they are then Gradient Boosting or Random Forest models post-fine tuning would be a good choice but would cost us interpretability and we would need to look out for overfitting.