Breast Cancer Detections from Tumor Traits

by Angelique Alexander

Introduction

Why?

This past summer, I was grabbing dinner with one of my close friends. She had a very serious look on her face and she said “Angelique I need you to stay calm, I have some news to share with you. I’m sorry I’m telling you a bit late, but it just hasn’t seemed like the right time to tell you every time we’ve met over the last couple of weeks.” Then she took a deep breath and said the words you never want to hear, “I have Breast Cancer.” I felt my chest tighten and my face fall.

My story is not that special or unique. My friend had a tumor that was benign for a long time and was regularly checked but became cancerous. Due to her diligence and the doctor’s close - monitoring, they were able to catch her breast cancer very early, Stage 1. Not everyone is as fortunate as me and my friend though.

This data has a very personal meaning behind it for me. Accurate detection of cancerous tumors (malignant) is so important in saving lives and early detection of cancer. That’s why I prioritized a high Recall score (meaning low false negatives) since that’s a new nightmare I understand the weight of.

Audience

This reading is meant for anyone that might be touched by breast cancer personally or through a loved one. It’s for anyone in their life with a woman they care about, a sister, a mother, a daughter, a friend.

This picture says everything, 1 in 8 women.

Data Source

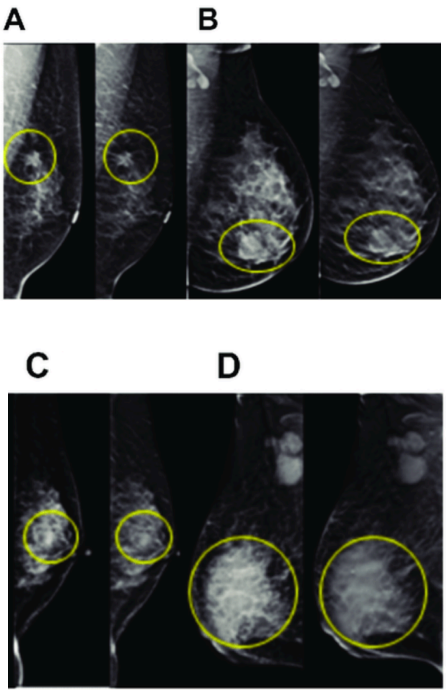
The Wisconsin dataset on Kaggle consists of 569 samples that are labelled

benign or malignant, with 30 features for each sample. Since this data set was meant for studies and was not presented with any predefined metrics for success, I created my own goals for looking at this data:

- Achieve an accuracy of 95% or higher in classifying malignant tumors, while considering other important metrics like Recall (lower false negatives) and Precision (lower false positives) to avoid misleading conclusions.
- Prioritize Recall to minimize false negatives (missing malignant tumors), ensuring the model captures as many positive cases as possible, while also maintaining an acceptable level of Precision.
- Perform feature selection to reduce the number of features and minimize redundancy, improving model interpretability and efficiency without sacrificing performance.

Data Wrangling

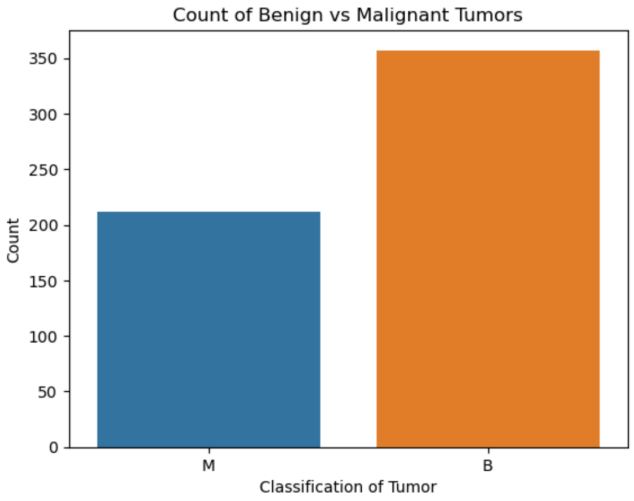
As someone who does not have expertise in the medical field, but does have a math background I was initially confused by some of the data. The first thing I noticed was the “radius_mean” value along with “radius_se” and “radius_worst”. This brought up a real question for me: How can there be more than one radius? This might seem obvious, but as a lay person it took some research to find out that tumors do not usually have a spherical shape. That means it might have different “radii” throughout the tumor where the “radius_mean” is the average of all the radii in a tumor and the “radius_worst” is the longest radius in the tumor. You can get an idea of the range in shapes for tumors with the images below.



(Chou & Pugh, 2019)
Tumors in different molecular subtypes of breast cancer. The yellow circles indicate the location of the lesion. (A) Luminal A subtype. Tumor size, 1.7x1.5 cm. (B) Luminal B subtype. Tumor size, 2.3 cm. (C) Triple-negative subtype. Tumor size, 1.4x1.0 cm. (D) Human epidermal growth factor receptor 2-overexpression subtype. Tumor size, 7x6 cm. The left images in each panel were obtained from full-field digital mammography and the right images in each panel were obtained from digital breast three-dimensional tomosynthesis.

Image source: https://www.researchgate.net/figure/Tumors-in-different-molecular-subtypes-of-breast-cancer-The-yellow-circles-indicate-the_fig1_330291005

The data for this project was already very clean with no missing values for any of the 569 samples. It didn't have any strange values in the dataset like 999 or other extreme values that were meant to substitute as values for unknown metrics. The only thing that really stands out as important in the data to take note of when wrangling it, is the imbalance of benign tumors vs malignant tumors as shown in the graph below:



In regards to the imbalance of data, I decided not to address it until later in the pipeline process.

Exploratory Data

In this phase I wanted to separate the data into two frames to see if there were any patterns within the benign vs malignant tumors. I created two separate data frames, one "b_df" and the other "m_df" to see if there might be different correlations between variables when we separated the categorization of the types of tumors.

There were a couple of columns from the original data that didn't seem to be helpful:

id', 'Unnamed: 32'. The 'ID' wasn't helpful since we weren't joining any data based on the ID's. The more important data was the category of Benign vs Malignant and the column 'Unnamed: 32' was filled only with NaN values so I dropped both of them.

I wanted to look at the summary of statistics on the remaining variables in my new data frames, so I used the .describe method to get the full summary statistics on the variables. This is what those generally looked like:

```
b_df.describe().T
```

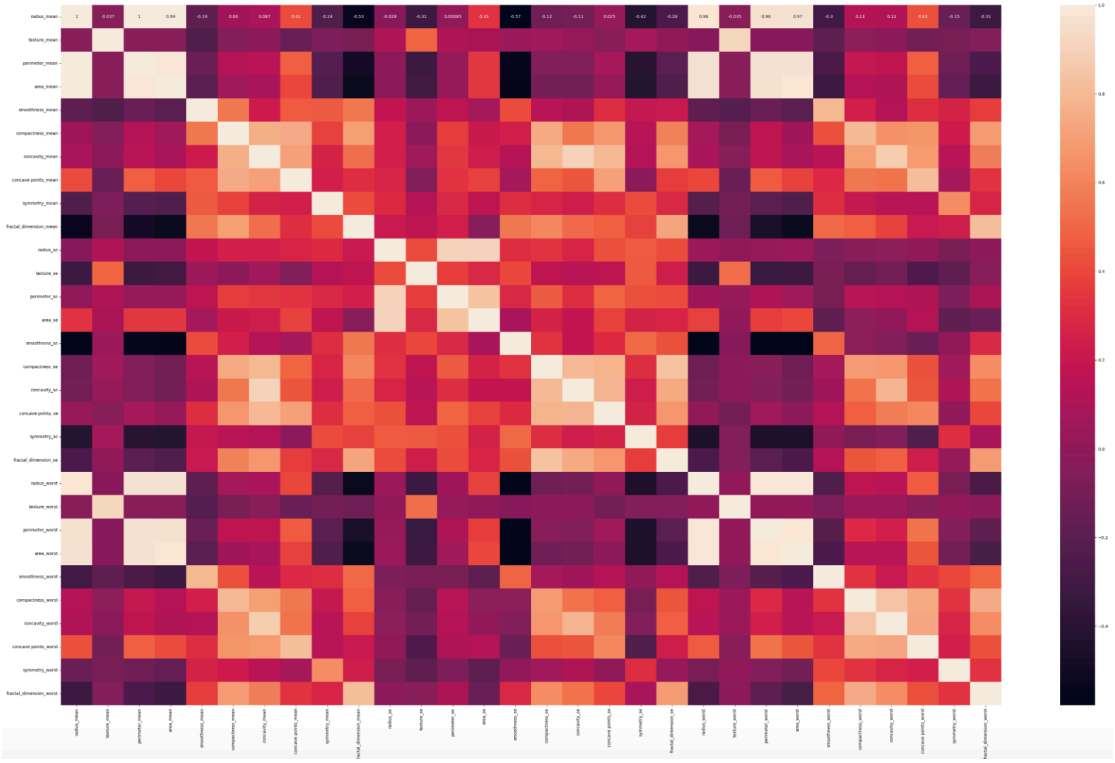
	count	mean	std	min	25%	50%	75%	max
radius_mean	357.0	12.146524	1.780512	6.981000	11.080000	12.200000	13.370000	17.85000
texture_mean	357.0	17.914762	3.995125	9.710000	15.150000	17.390000	19.760000	33.81000
perimeter_mean	357.0	78.075406	11.807438	43.790000	70.870000	78.180000	86.100000	114.60000
area_mean	357.0	462.790196	134.287118	143.500000	378.200000	458.400000	551.100000	992.10000
smoothness_mean	357.0	0.092478	0.013446	0.052630	0.083060	0.090760	0.100700	0.16340
compactness_mean	357.0	0.080085	0.033750	0.019380	0.055620	0.075290	0.097550	0.22390
concavity_mean	357.0	0.046058	0.043442	0.000000	0.020310	0.037090	0.059990	0.41080
concave points_mean	357.0	0.025717	0.015909	0.000000	0.015020	0.023440	0.032510	0.08534
symmetry_mean	357.0	0.174186	0.024807	0.106000	0.158000	0.171400	0.189000	0.27430
fractal_dimension_mean	357.0	0.062867	0.006747	0.051850	0.058530	0.061540	0.065760	0.09575
radius_se	357.0	0.284082	0.112570	0.111500	0.207300	0.257500	0.341600	0.88110
texture_se	357.0	1.220380	0.589180	0.360200	0.795900	1.108000	1.492000	4.88500
perimeter_se	357.0	2.000321	0.771169	0.757000	1.445000	1.851000	2.388000	5.11800
area_se	357.0	21.135148	8.843472	6.802000	15.260000	19.630000	25.030000	77.11000
smoothness_se	357.0	0.007196	0.003061	0.001713	0.005212	0.006530	0.008534	0.02177
compactness_se	357.0	0.021438	0.016352	0.002252	0.011320	0.016310	0.025890	0.10640
concavity_se	357.0	0.025997	0.032918	0.000000	0.010990	0.018400	0.030560	0.39600
concave points_se	357.0	0.009858	0.005709	0.000000	0.006433	0.009061	0.011870	0.05279
symmetry_se	357.0	0.020584	0.006999	0.009539	0.015600	0.019090	0.024060	0.06146
fractal_dimension_se	357.0	0.003636	0.002938	0.000895	0.002074	0.002808	0.004174	0.02984

```
m_df.describe().T
```

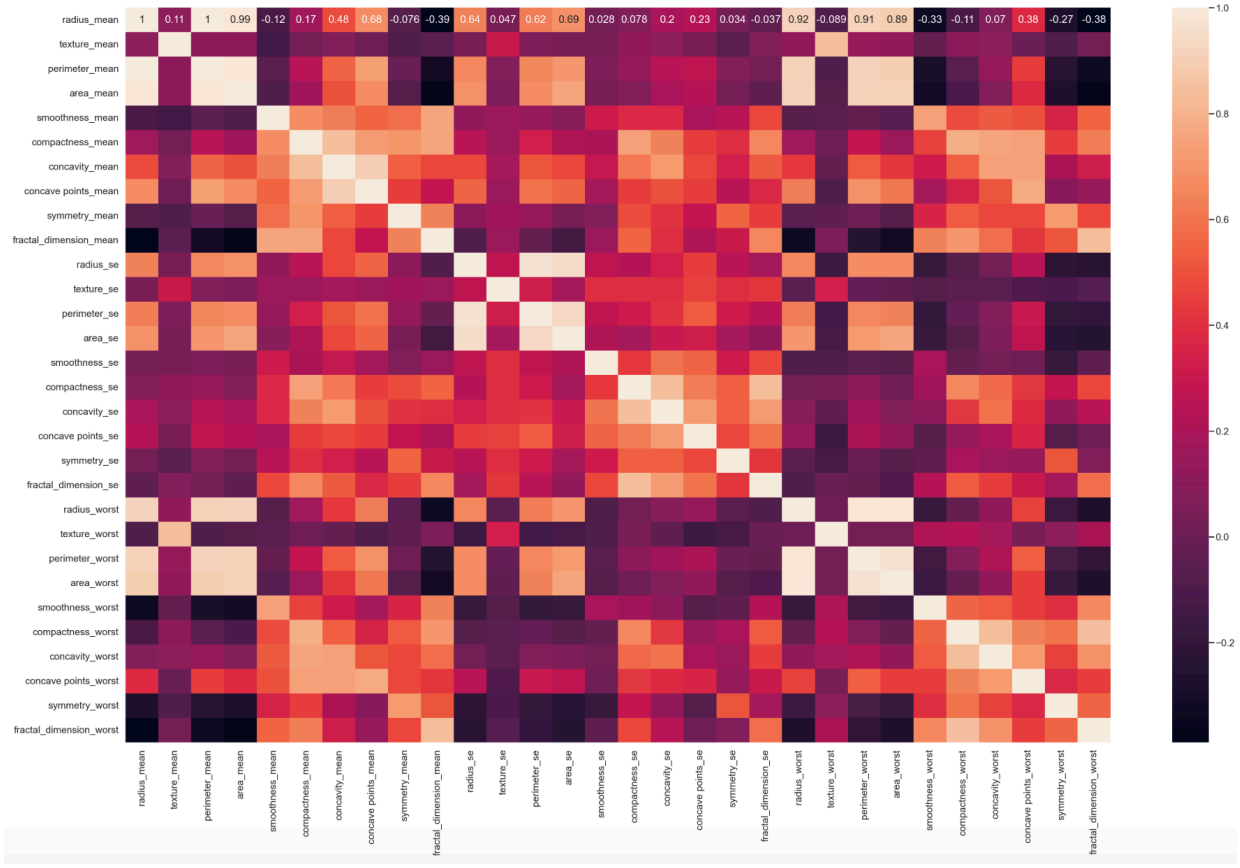
	count	mean	std	min	25%	50%	75%	max
radius_mean	212.0	17.462830	3.203971	10.950000	15.075000	17.325000	19.590000	28.11000
texture_mean	212.0	21.604906	3.779470	10.380000	19.327500	21.460000	23.765000	39.28000
perimeter_mean	212.0	115.365377	21.854653	71.900000	98.745000	114.200000	129.925000	188.50000
area_mean	212.0	978.376415	367.937978	361.600000	705.300000	932.000000	1203.750000	2501.00000
smoothness_mean	212.0	0.102898	0.012608	0.073710	0.094010	0.102200	0.110925	0.14470
compactness_mean	212.0	0.145188	0.053987	0.046050	0.109600	0.132350	0.172400	0.34540
concavity_mean	212.0	0.160775	0.075019	0.023980	0.109525	0.151350	0.203050	0.42680
concave points_mean	212.0	0.087990	0.034374	0.020310	0.064620	0.086280	0.103175	0.20120
symmetry_mean	212.0	0.192909	0.027638	0.130800	0.174050	0.189900	0.209850	0.30400
fractal_dimension_mean	212.0	0.062680	0.007573	0.049960	0.056598	0.061575	0.067075	0.09744
radius_se	212.0	0.609083	0.345039	0.193800	0.390375	0.547200	0.757300	2.87300
texture_se	212.0	1.210915	0.483178	0.362100	0.892825	1.102500	1.429250	3.56800
perimeter_se	212.0	4.323929	2.568546	1.334000	2.715500	3.679500	5.206250	21.98000
area_se	212.0	72.672406	61.355268	13.990000	35.762500	58.455000	94.000000	542.20000
smoothness_se	212.0	0.006780	0.002890	0.002667	0.005085	0.006209	0.007971	0.03113
compactness_se	212.0	0.032281	0.018387	0.008422	0.019662	0.028590	0.038910	0.13540
concavity_se	212.0	0.041824	0.021603	0.011010	0.026998	0.037125	0.050443	0.14380
concave points_se	212.0	0.015060	0.005517	0.005174	0.011415	0.014205	0.017497	0.04090
symmetry_se	212.0	0.020472	0.010065	0.007882	0.014615	0.017700	0.022132	0.07895
fractal_dimension_se	212.0	0.004062	0.002041	0.001087	0.002688	0.003739	0.004892	0.01284

I also thought that the variables correlation heatmaps might look different for the benign vs malignant tumors. My thinking there was that benign tumors might have different variables with stronger correlation than the malignant tumors did. There were slight differences, but overall they were very similar.

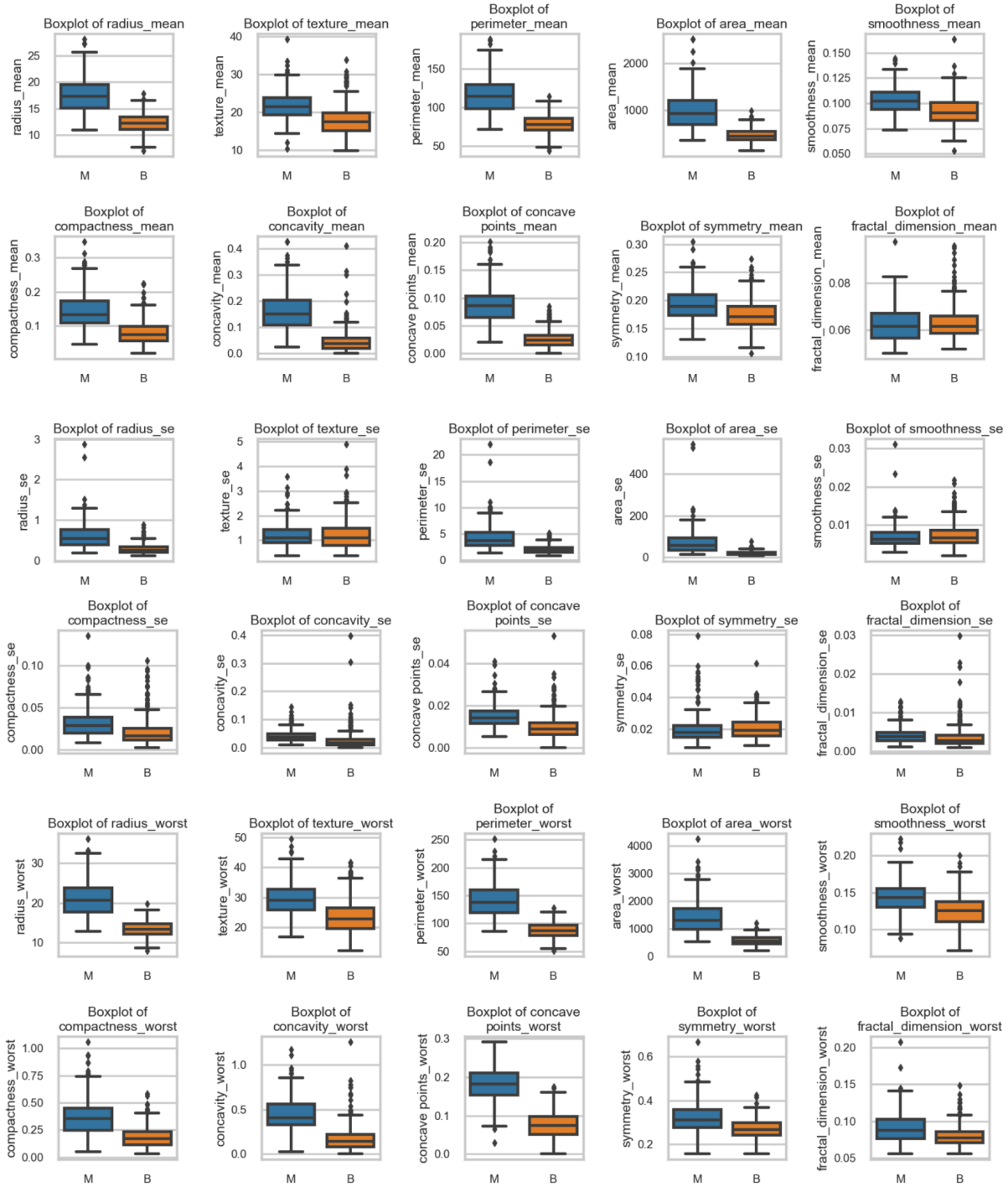
Benign Heatmap



Malignant Heatmap



The other thing I wanted to be careful to check for was outliers in the data. The way I looked for this was by using boxplots comparing the benign and malignant data.



The standard error (SE) metrics, marked by _se, show significant outliers in both benign and malignant tumors. This is expected, as both tumor types have a wide range of values across features, which can cause outliers in the SE calculations.

This suggests that the outliers in both the mean and SE metrics are likely a natural part of the data, rather than errors. If the SE didn't show similar outliers, I would reconsider keeping rows with extreme mean outliers. However, since both metrics show outliers, removing rows was not necessary.

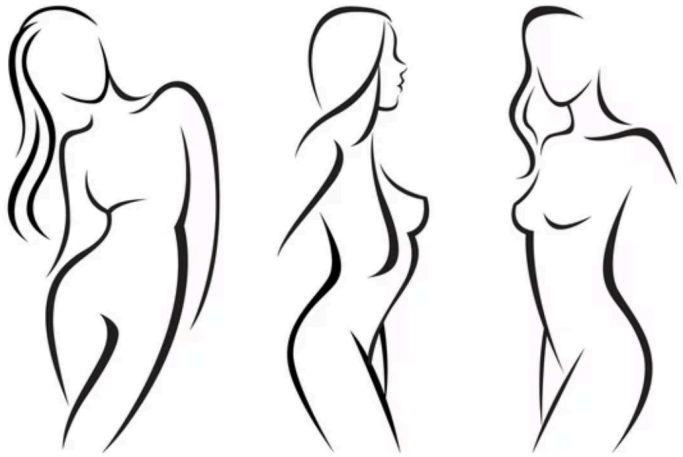
Pre-processing and Training

In this section I looked for variables that were redundant in the data due to high correlation. The mean and worst variables had the highest correlations so I started with another heatmap to look for correlation then converted it to a readable matrix version:

	Feature 1	Feature 2	Correlation
8	radius_mean	radius_worst	0.969539
69	concavity_mean	concave points_mean	0.921391
25	texture_mean	texture_worst	0.912045
93	concave points_mean	concave points_worst	0.910155
188	compactness_worst	concavity_worst	0.892261
76	concavity_mean	concavity_worst	0.884103
52	compactness_mean	concavity_mean	0.883121
59	compactness_mean	compactness_worst	0.865809
77	concavity_mean	concave points_worst	0.861323
205	concavity_worst	concave points_worst	0.855434
53	compactness_mean	concave points_mean	0.831135
88	concave points_mean	radius_worst	0.830318

Based on this I dropped some more variables and ultimately ended up reducing it to 18 variables. I changed over the classification of Benign and Malignant to numeric values, 0 and 1 respectively. From there I tried several different models to fit the data and used a function to measure each models accuracy, precision, recall and ROC AUC scores. This gave me a good start, but I really refined the models effectiveness in my modeling notebook.

Modeling



I oversampled the data to address the class imbalance because the malignant tumors (the minority class) were underrepresented. This step helped ensure that the model was trained on a more balanced dataset, which can improve its predictive performance. After oversampling, I set the random_state = 88 to ensure that the results were replicable across different runs. Additionally, I used cross-validation to evaluate the models' performance, allowing me to assess how well they generalize to unseen data by training and testing on different subsets of the data.

1. Logistic Regression: Logistic regression is a powerful algorithm for binary classification tasks, which is ideal for the tumor diagnosis problem in this dataset. There are several types of logistic regression models: binomial, multinomial, and ordinal. Since our target variable (the tumor diagnosis) has two possible outcomes — malignant or benign — the binomial logistic regression model is the most suitable choice.

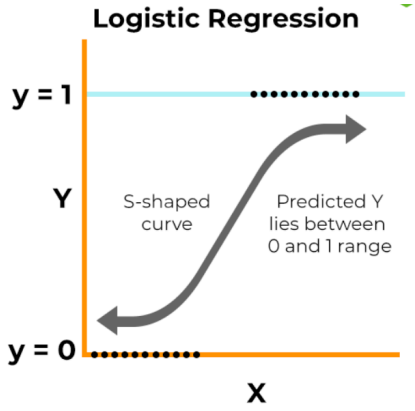


Image Source: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/

To optimize the model, I used the GridSearchCV to tune the hyper-parameters. The best hyper-parameters found were:

- C = 10 This value controls the regularization strength. A higher value (like 10) indicates less regularization, allowing the model to fit the data more flexibly, which is useful when trying to capture complex patterns.
- Solver = 'liblinear': This solver is designed for smaller datasets and binary classification tasks. It works efficiently and is particularly suited for the logistic regression model in this case.

The performance metrics from the logistic regression model are as follows:

- Accuracy: 97.67%
- Precision: 97%
- Recall: 97.98%
- ROC AUC: 97.70%
- Cross-validation score: 95.59%

These results suggest that the logistic regression model performs well in distinguishing between benign and malignant tumors, with high accuracy and recall rates, making it an effective classifier for this problem.

2. Random Forest Classifier: The Random Forest Classifier is an ensemble learning method that combines the predictions of multiple decision trees. Each tree in the forest makes an independent prediction, and the final model's prediction is determined by the majority vote from all the individual trees. This approach helps improve the accuracy and robustness of predictions by reducing the likelihood of overfitting, which is common

in individual decision trees. Random Forest can be used for both classification and regression tasks, making it a versatile algorithm for various types of data.

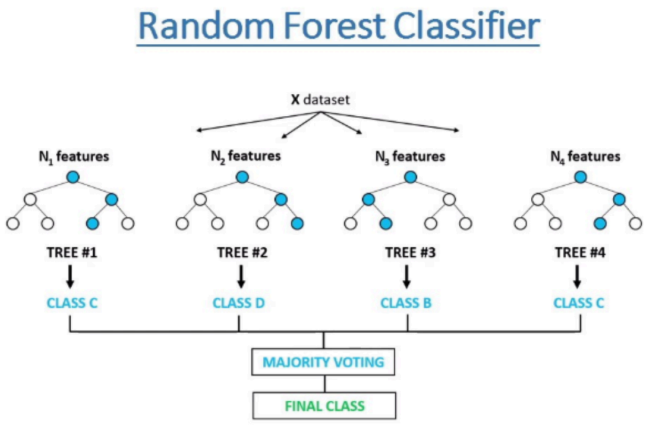


Image Source: https://www.freecodecamp.org/news/how-to-use-the-tree-based-algorithm-for-machine-learning/

To optimize the Random Forest model, I used RandomizedSearchCV to perform a randomized search for the best hyper-parameters. The optimal hyper-parameters found were:

- 'n_estimators' = 200: This is the number of decision trees in the forest. A higher value typically improves model accuracy but increases computational cost. I chose 200 trees for a good balance between performance and computational efficiency.
- 'min_samples_split' = 5: This parameter controls the minimum number of samples required to split an internal node. A value of 5 helps prevent overfitting by ensuring that nodes are split only when there are enough samples.
- 'min_samples_leaf' = 1: This parameter sets the minimum number of samples that a leaf node must have. A value of 1 allows the tree to grow deeper, capturing more detailed patterns, but can also increase the risk of overfitting if set too low.
- 'max_features' = 'sqrt': This specifies the number of features to consider when looking for the best split. Using the square root of the total number of features helps improve model performance and prevents overfitting by limiting the amount of information each tree can use.
- 'max_depth' = 10: The maximum depth of the trees. Limiting the depth prevents overfitting by ensuring that the trees do not grow too complex.
- 'bootstrap' = True: This indicates that the trees are built using bootstrap sampling, which means each tree is trained on a random subset of the data with replacement. This technique helps improve model generalization.

The performance of the Random Forest Classifier model was as follows:

- Accuracy: 95.35%
- Precision: 96.84%
- Recall: 92.93%
- ROC AUC: 95.17%
- Cross-validation score: 94.79%

These results demonstrate that the Random Forest Classifier performs well in classifying the data, with high precision, recall, and accuracy. The model effectively balances avoiding false positives and minimizing false negatives, making it a reliable classifier for this task. However, its performance is slightly lower than that of the Logistic Regression model, which achieved stronger results.

3. K Nearest Neighbor: The K-Nearest Neighbors algorithm is based on the idea that similar points tend to be close to each other. If two data points are in the same group, they are likely to appear near one another in the feature space. The key parameter in KNN is k, which defines the number of nearest neighbors to consider when making a prediction. The smaller the distance between the points, the more likely they are to belong to the same class.

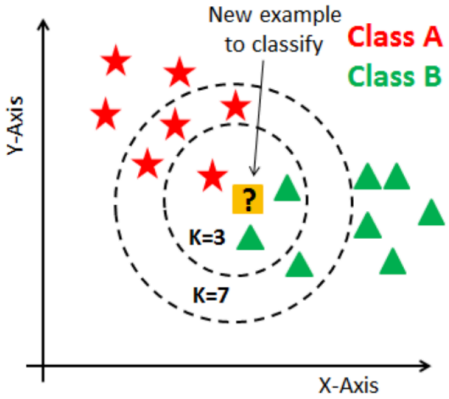


Image Source: https://ai.plainenglish.io/introduction-to-k-nearest-neighbors-knn-algorithm-e8617a448fa8

To optimize the KNN model, I used RandomizedSearchCV to perform a randomized search for the best hyper-parameters. The optimal values found were:

- algorithm: 'brute' : (indicating the use of a brute-force search to compute distances)
- n_neighbors: 6 : (the number of nearest neighbors to consider)
- p: 2 : (indicating the use of the Euclidean distance, as p=2 corresponds to the standard Euclidean metric)
- weights: 'distance' : (meaning the influence of each neighbor is weighted by its inverse distance)

The results from the KNN model were:

- Accuracy: 94.88%
- Precision: 92.31%
- Recall: 96.97%
- ROC AUC: 95.04%
- Cross Validation Score: 96.39%

Overall, the KNN model seems to be effective in classifying the tumors as either malignant or benign, with a good balance of precision and recall. It would be a useful model for breast cancer detection, but might be slightly less performant than other models (like Logistic Regression, based on your earlier results).

4. **Support Vector Classifier:** The Support Vector Classifier (SVC) is a robust machine learning algorithm that excels at classification tasks, particularly when dealing with high-dimensional or non-linear data, which is the case in this problem. It works by identifying the optimal decision boundary (hyperplane) that maximizes the margin between different classes. SVC can handle complex datasets by utilizing the kernel trick, which maps data into a higher-dimensional space to make non-linear separations feasible. With careful tuning of key hyper-parameters like C and the choice of kernel, SVC can yield excellent performance across a variety of classification problems.

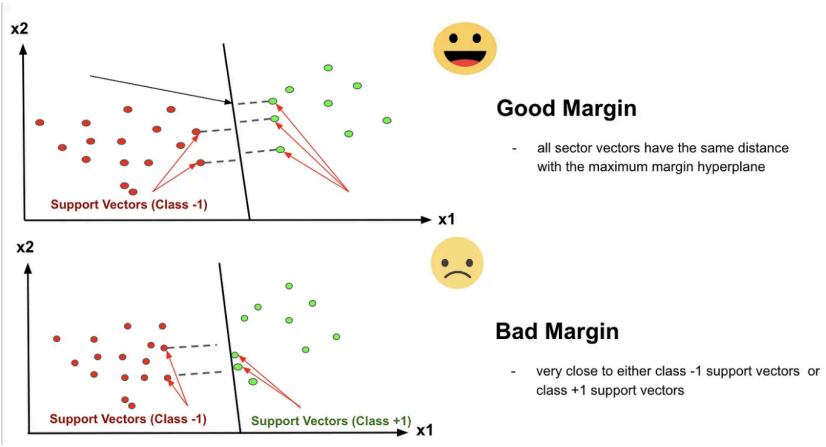


Image Source: A Top Machine Learning Algorithm Explained: Support Vector Machines (SVMs)

To optimize the SVC model for this dataset, I used RandomizedSearchCV to search for the best hyper-parameter combination. The optimal values found were:

- Kernel: 'rbf' : (Radial Basis Function)
- Gamma: 'scale' : (which adjusts the gamma value based on the number of features)
- Degree: 4 : (the degree of the polynomial kernel function, relevant if using a polynomial kernel)
- Coef0: 0 : (a constant added to the kernel function, relevant for polynomial and sigmoid kernels)
- C: 10 : (controls the trade-off between margin size and classification errors)

The performance of the optimized SVC model was as follows:

- Accuracy: 96.74%
- Precision: 96.00%
- Recall: 96.97%
- ROC AUC: 96.76%
- Cross-Validation Score: 94.79%

These results demonstrate that the SVC model performs effectively in classifying the data, achieving high precision, recall, and accuracy. The model is able to balance correctly classifying positive cases (malignant tumors) while maintaining a low rate of false positives (benign tumors). However, despite the strong performance, the results are slightly lower than those obtained from the Logistic Regression model, which achieved even better classification metrics.

5. **Decision Tree Classifier:** The Decision Tree Classifier is an intuitive and versatile model that works by recursively splitting the data into subsets based on feature values to achieve the best class separation. It is easy to visualize and interpret, making it a popular choice for classification tasks. A key strength of decision trees is that they can handle both numerical and categorical data. However, they are prone to overfitting, especially when the tree grows too deep. To mitigate this, pruning techniques (such as limiting the depth of the tree or requiring a minimum number of samples to split) can be applied to enhance generalization.

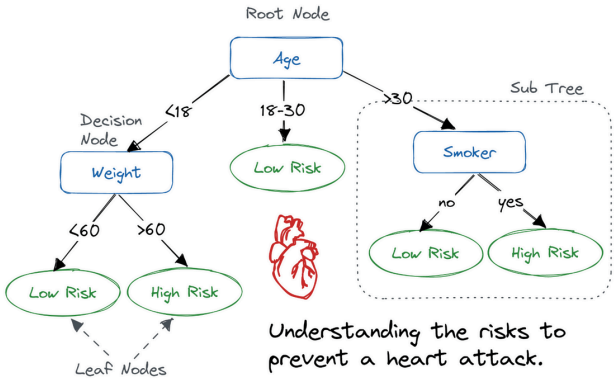


Image | Abid Ali Awan

In this case, I used the GridSearchCV to fine-tune the hyper-parameters for the Decision Tree Classifier. The best combination of hyper-parameters found was:

- Criterion: 'gini' : (used to measure the quality of a split)
- Max Depth: 5 : (limits the depth of the tree to prevent overfitting)
- Max Features: None : (all features are considered for splitting)
- Min Samples Leaf: 4 : (the minimum number of samples required at a leaf node)
- Min Samples Split: 10 : (the minimum number of samples required to split an internal node)
- Random State: 42 : (ensures reproducibility of results)

The performance of the optimized Decision Tree Classifier model was as follows:

- Accuracy: 0.9302
- Precision: 0.9375
- Recall: 0.9091
- ROC AUC: 0.9287
- Cross-validation score: 93.79

These results show that the Decision Tree model performs well, with solid precision, recall, and accuracy. However, despite these positive metrics, the performance is slightly lower compared to all the other models, especially the Logistic Regression and SVC models.

Conclusion:

	Accuracy	Recall	Precision	Cross Val Avg
Logistic Regression	0.976744	0.979798	0.970000	95.587879
Random Forest	0.953488	0.929293	0.968421	94.789899
KNN	0.948837	0.969697	0.923077	96.387879
SVM	0.967442	0.969697	0.960000	94.787879
Decision Tree	0.930233	0.909091	0.937500	93.787879

This report is designed to be accessible to anyone affected by breast cancer, regardless of their familiarity with machine learning. While I sought additional data to enhance the dataset, unfortunately, no publicly available datasets were found to expand the current data. Through this analysis, I aimed to walk readers through the data and explore the various machine learning models that can be used to predict whether a tumor is malignant or benign.

After evaluating multiple models, I concluded that the Logistic Regression model performs the best. It not only provides accurate predictions but also minimizes both false positives and false negatives. This is crucial in a medical context, as reducing false negatives (missed malignant tumors) and false positives (misidentified benign tumors) can directly contribute to saving lives.