Breast Cancer Diagnosis Using Ensemble Learning Methods

Sunny Sharma

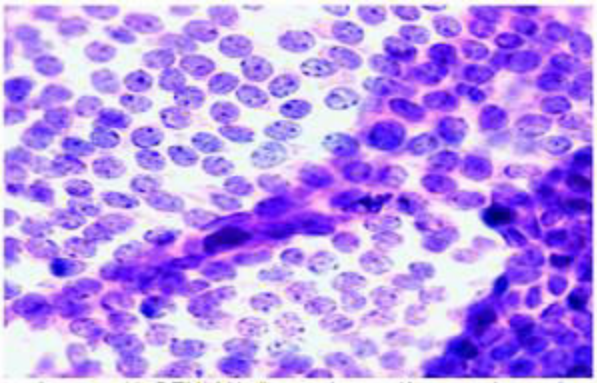
Definition

Project Overview

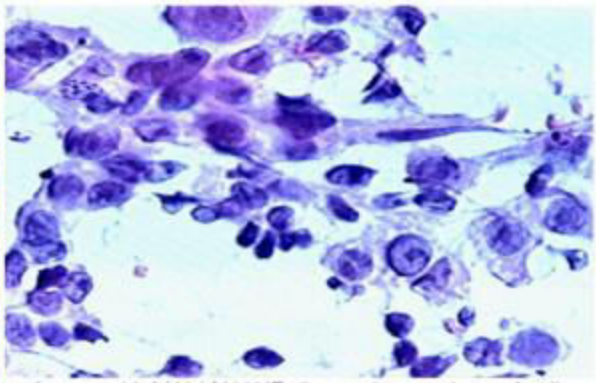
According to the American Cancer Society, roughly 231,840 females in the United States suffered from invasive cases of breast cancer in 2015. These cases resulted in about 40,290 deaths[1]. Early diagnosis of breast cancer is the key to effective treatment of the disease. By leveraging machine learning methods for medical diagnostics, valuable insights into the nature and treatment of all cancers including breast cancer can be made, and these insights can save thousands of lives annually.

In this project, we use ensemble learning to understand characteristics of cell nuclei and from these characteristics, classify the cells as either benign or malignant. The characteristics of the cell nuclei are obtained from digital images of a fine needle aspirate (FNA) of breast mass. Fine needle aspiration is a biopsy procedure in which a thin hollow needle is inserted into the tissue in question to obtain a sample of cells which are stained and then can be studied under a microscope. This problem and dataset can be found on kaggle. Examples of benign and malignant FNA samples of breast tissue are pictured below.

[2]



Smear with BENIGN diagnosis – uniform nucleus of cells, symmetrical, homogeneous, with areas within normal size



Smear with MALIGNANT diagnosis – nucleus of cells without uniformity, asymmetrical, not homogeneous (multiple sizes) and with areas above normal size

For each of the 569 instances in the dataset these FNA images were inspected and 10 values were calculated for each individual cell nucleus found in the picture. These 10 values are listed below:

1. Radius
2. Texture (standard deviation of gray-scale values)
3. Perimeter
4. Area
5. Smoothness (local variation in radius lengths)
6. Compactness (perimeter^2 / area – 1.0)
7. Concavity (severity of concave portions of the contour)
8. Concave points (number of concave portions of the contour)
9. Symmetry
10. Fractal dimension (“coastline approximation” – 1)

These values for all cells pictured were then aggregated and the means, the standard errors, and the worst values (mean of 3 largest values) of each were calculated to produce 30 values. These 30 features are the input space of the problem. The raw data, made available by the University of Wisconsin (and downloaded from kaggle[3]), also includes the id of each of the 569 patients, and the actual diagnosis, or true class label, of either benign or malignant.

Problem Statement

To solve this supervised classification task, random forests were used. The method of random forests involves using an ensemble of decision trees. When building a single decision tree for classification, one would want to choose the branch splits to maximize purity or information gain of the distribution of class label conclusions for the training data on the leaves on either side. When using random forests for classification however, rather than choosing splits based on maximizing purity for the entirety of the training data, each of the trees are built using a randomly chosen subset of training samples. The ultimate classification conclusion of the random forest is a vote between all of the decision trees built. Using random subsets of the training data instead of the entire training set to build many trees instead of just one is useful because it increases the bias of each of the trees and decreases the variance, which helps to protect against overfitting to the training data and a lack of ability to generalize well to unseen data, which is a big weakness of single decision trees. We can take this randomness a step further in a method known as extremely randomized trees, where random subsets of the training data are used to build the trees but rather than splitting branches for maximum purity, thresholds are drawn at random for each feature and the best random split is picked to build the tree with. This allows for a further decrease in variance and increase in bias. The solutions of both random forests and extremely randomized trees will be explored and tuned to find the best predictive model for classification, as well as a simple decision tree for some context as a benchmark.

Metrics

The metric we will use to determine which of these models is truly the best solution is the f1 score, or a harmonic mean of precision and recall. The equations for the f1 score, precision and recall are given below:

$$F_1 = 2\frac{precision \times recall}{precision + recall}$$

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

Precision can be thought of as a measure of exactness of our models and recall can be thought of as a measure of completeness. The reason we opt for the F_1 score comprised of precision and recall instead of using simple accuracy as a metric is that we want a metric which takes into account the completeness of classifications seeing as how although not greatly imbalanced our dataset is slightly imbalanced as we'll see in the following section.

Analysis

Data Exploration

Before beginning to implement the aforementioned solution, it is worthwhile to explore the data. Below lies a small random sample of some instances in the dataset.

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
1	842517	M	20.570	17.77	132.90	1326.0	0.08474	0.07864	0.08690
182	873701	M	15.700	20.31	101.20	766.6	0.09597	0.08799	0.06593
412	905539	B	9.397	21.68	59.75	268.8	0.07969	0.06053	0.03735
433	908445	M	18.820	21.97	123.70	1110.0	0.10180	0.13890	0.15940
357	901028	B	13.870	16.21	88.52	593.7	0.08743	0.05492	0.01502
222	8812844	B	10.180	17.53	65.12	313.1	0.10610	0.08502	0.01768

concave points_mean	symmetry_mean	fractal_dimension_mean	radius_se	texture_se	perimeter_se	area_se	smoothness_se	compactness_se
0.070170	0.1812	0.05667	0.5435	0.7339	3.398	74.080	0.005225	0.013080
0.051890	0.1618	0.05549	0.3699	1.1500	2.406	40.980	0.004626	0.022630
0.005128	0.1274	0.06724	0.1186	1.1820	1.174	6.802	0.005515	0.026740
0.087440	0.1943	0.06132	0.8191	1.9310	4.493	103.900	0.008074	0.040880
0.020880	0.1424	0.05883	0.2543	1.3630	1.737	20.740	0.005638	0.007939
0.019150	0.1910	0.06908	0.2467	1.2170	1.641	15.050	0.007899	0.014000

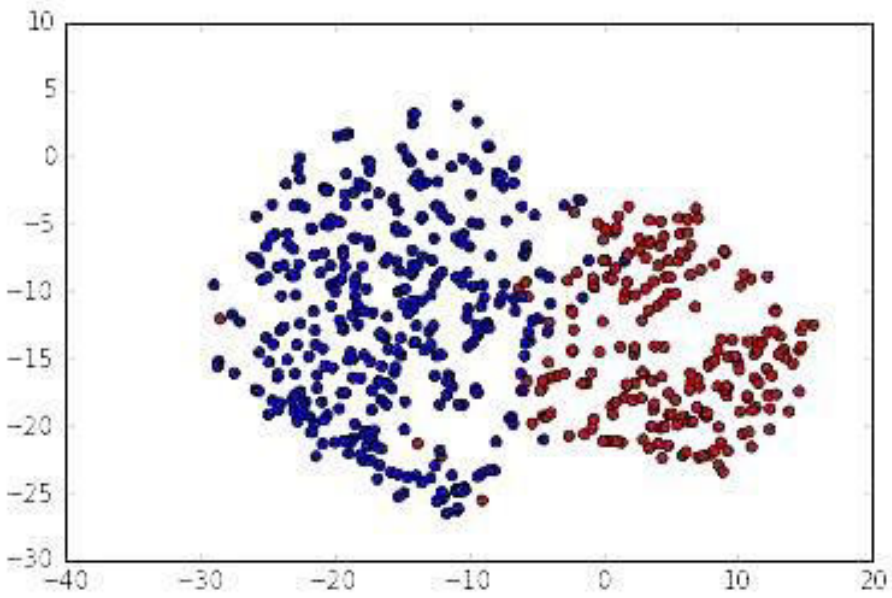
concavity_se	concave points_se	symmetry_se	fractal_dimension_se	radius_worst	texture_worst	perimeter_worst	area_worst	smoothness_worst
0.018600	0.013400	0.01389	0.003532	24.990	23.41	158.80	1956.0	0.1238
0.019540	0.009767	0.01547	0.002430	20.110	32.82	129.30	1269.0	0.1414
0.037350	0.005128	0.01951	0.004583	9.965	27.99	66.61	301.0	0.1086
0.053210	0.018340	0.02383	0.004515	22.660	30.93	145.30	1603.0	0.1390
0.005254	0.006042	0.01544	0.002087	15.110	25.58	96.74	694.4	0.1153
0.008534	0.007624	0.02637	0.003761	11.170	22.84	71.94	375.6	0.1406

compactness_worst	concavity_worst	concave points_worst	symmetry_worst	fractal_dimension_worst
0.1866	0.24160	0.18600	0.2750	0.08902
0.3547	0.29020	0.15410	0.3437	0.08631
0.1887	0.18680	0.02564	0.2376	0.09206
0.3463	0.39120	0.17080	0.3007	0.08314
0.1008	0.05285	0.05556	0.2362	0.07113
0.1440	0.06572	0.05575	0.3055	0.08797

When exploring this data to get a better understanding of it, the first thing to note is the class distribution, shown below.

Class distribution of Breast Cancer Wisconsin (Diagnostic) data set		
Class	**# of instances**	**% of dataset**
Benign	357	62.74
Malignant	212	37.26
Total	569	

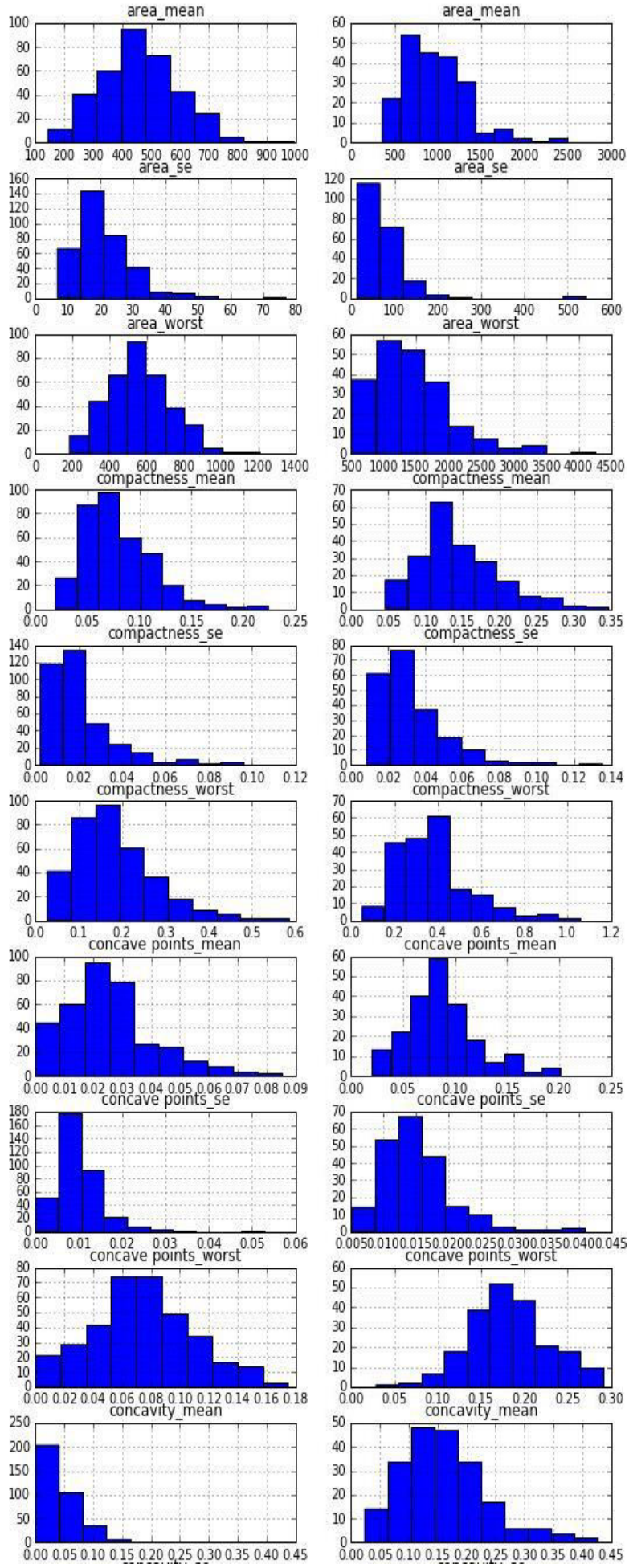
Before visualizing individual features of this dataset, let's see if we can get an idea of how the data is distributed as a whole. One way to visualize high dimensional data in order to get an idea of its distribution is through manifold learning, specifically t-distributed stochastic neighbor embedding, or t-SNE. T-SNE is an algorithm for nonlinear dimensionality reduction that can embed high-dimensional data into two dimensional spaces, allowing us to represent high-dimensional data with a simple scatterplot. T-SNE works by constructing a probability distribution over pairs of instances in the high-dimensional space so that probabilities are proportional to the similarities of the instances. A similar probability distribution is made for the pairs of instances in the lower dimensional space, and the difference between these distributions is minimized using gradient descent so that, ultimately, objects that are closer together, or more similar, in the high-dimensional input space will be closer together in the lower dimensional output space. The result of applying t-SNE to the dataset is shown below:

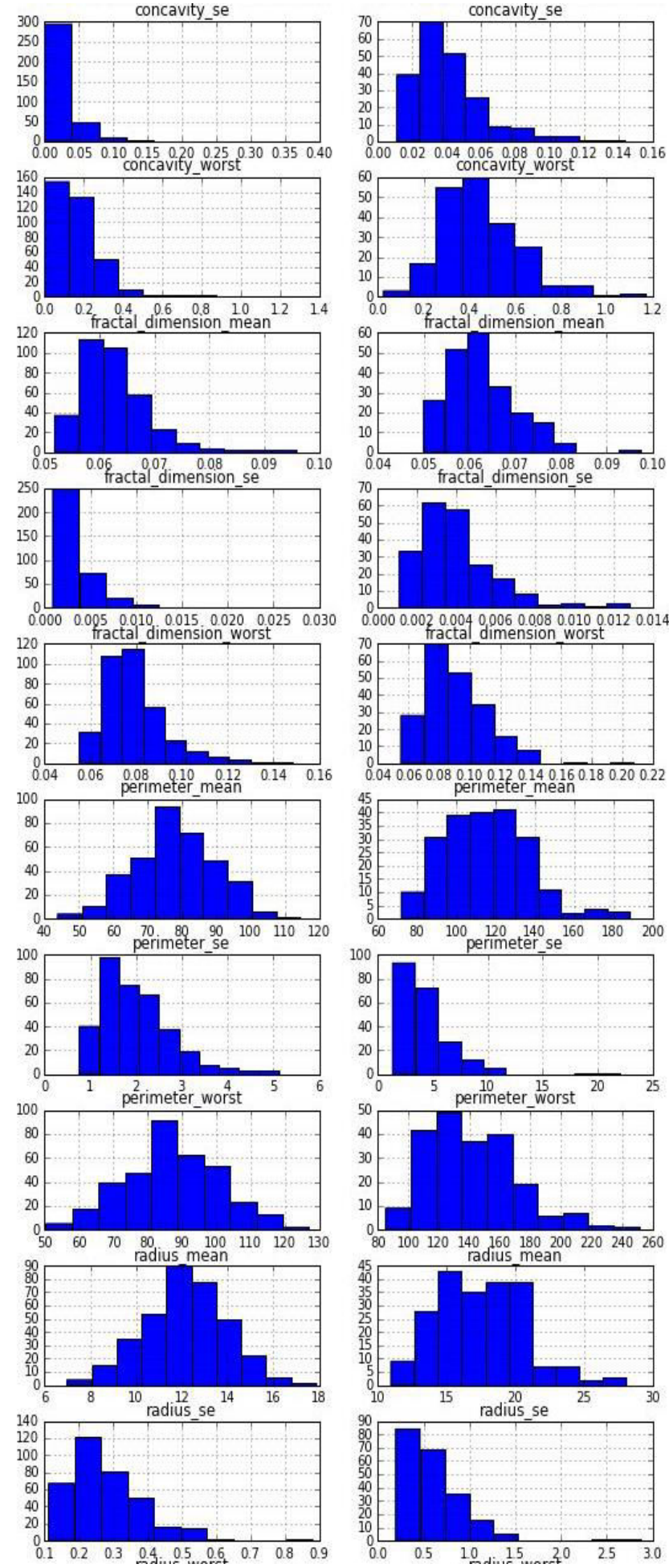


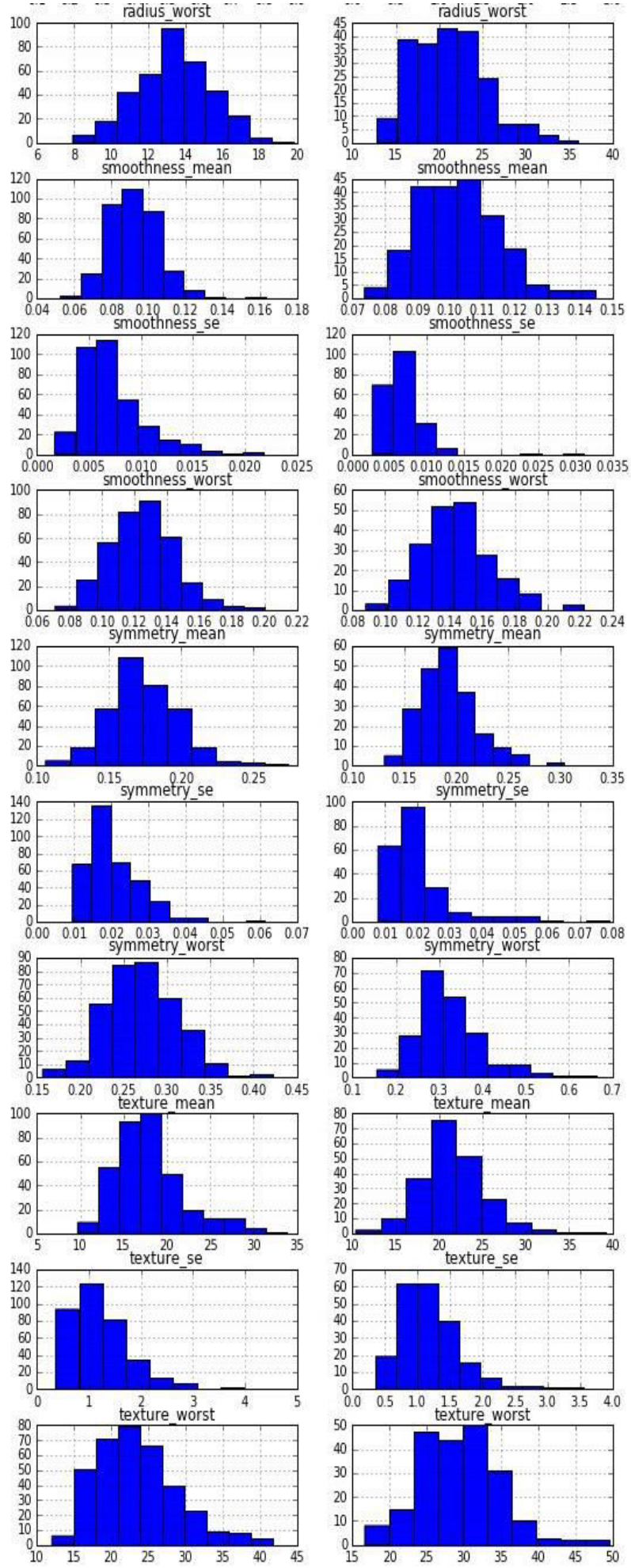
In this visualization, the benign diagnoses are blue while the malignant diagnoses are red. There is a clear cluster of each, hinting that there are definite trends to pick up on which will allow for the classification of the data.

Exploratory Visualization

A useful way to get some insight into each feature present in the data is to look at histograms of all 30 features separated by diagnosis and see if any rough distinctions become clear just by eyeballing the distributions of data. The histograms for each diagnosis are shown below with benign on the left and malignant on the right of each column:


area_mean
area_mean
area_se
area_se
area_worst
area_worst
compactness_mean
compactness_mean
compactness_se
compactness_se
compactness_worst
compactness_worst
concave points_mean
concave points_mean
concave points_se
concave points_se
concave points_worst
concave points_worst
concavity_mean
concavity_mean


concavity_se
concavity_se
concavity_worst
concavity_worst
fractal_dimension_mean
fractal_dimension_mean
fractal_dimension_se
fractal_dimension_se
fractal_dimension_worst
fractal_dimension_worst
perimeter_mean
perimeter_mean
perimeter_se
perimeter_se
perimeter_worst
perimeter_worst
radius_mean
radius_mean
radius_se
radius_se



When comparing the distributions of variables by diagnosis, we see many differences both in the shape and scale of distributions. However, the discrepancies for some variables stand out more than others. Specifically, “concave points_worst”, where the distributions for both diagnoses look normal but the benign is centered around .07 while the malignant is centered around .17, and “radius_worst”, where benign diagnoses look normally distributed about 13 while malignant diagnoses look skewed right with a very small fraction of instances between 13 and 15. The large discrepancies of these and several other features could represent efficient places to split our decision trees in order to accurately classify our data.

Algorithms and Techniques

With a better grasp on the dataset, it’s time to begin implementing a solution. As mentioned before, to solve this classification problem we will use an ensemble of decision

trees, each built from a random subset of the training data so that each split based on the features maximizes purity on either side, a random forest classifier, as well as an ensemble of decision trees built from random subsets of the training data so that each split represents a randomly chosen split, among many similarly randomly generated splits, that happens to separate the data the best, for an extremely randomized trees classifier.

Both of these algorithms use a similar set of parameters which play an important role in the ultimate performance of the classifiers. Some key parameters include the number of estimators, or the number of decision trees to be constructed and consulted to classify an instance of data, the maximum amount of features to consider when looking for the best split, and the minimum samples required on either branch side to allow for a node to be split. These are among the parameters that will be tweaked to achieve a fine-tuned and improved classifier after a basic version has been implemented.

Some reasons why we use randomized forests and extremely randomized trees include that they are some of the most accurate classification algorithms available, they work well with high dimensional input spaces, they also work well with unbalanced datasets, and in utilizing these algorithms, we can get a sense of which input variables are more important to consider to classify data instances correctly by looking at which features the trees split on and how. Understanding the splits of our trees can give us insight into the nature of the problem and solution. There are a couple pitfalls of these algorithms that we will have to deal with as well. Firstly they are not as easily interpretable by humans as a single decision tree. Furthermore they can give us an idea of the importance of features, but when working with an input space of features of which some might be correlated, this notion of importance can get muddled or

suppressed among some of them. Finally, these algorithms can be prone to overfitting in the case of noisy data, however this shortcoming is mitigated by the fact that our input space is made up of aggregated statistics, which controls for the effects of noise.

A couple reasons that we use decision trees, random forests, and extremely random forests specific to our dataset is that we have a 30 dimensional input space where some features are bound to be very relevant to the classification task while some features might be utterly irrelevant, and decision trees and ensemble decision tree learning methods handle datasets where relevance can vary wildly from feature to feature rather well. Furthermore, with decision trees and ensemble decision tree learning, we can learn exactly which features are the most important in the classification task, which teaches us about the nature of malignant cell nuclei while many other classification algorithms have no concept of feature importance that we can learn about the problem from.

Benchmark

Understanding our classifiers performance requires some context, so to act as a benchmark, we will use a simple decision tree's average f1 score when classifying our dataset with 3-fold cross validation. Below is a table showing this performance benchmark as well as a simple majority vote and support vector classifier for further context.

Performance benchmarks	
Method	F1 score
Decision tree	0.9213
Simple majority vote (always predict benign)	0.7711
Support Vector Classifier	0.77106

Methodology

Data Preprocessing

No preprocessing was done on the data for two reasons, the first of which being that randomized forests and decision tree algorithms in general work well with a variety of data whether it is all of the same scale or not or part categorical and part numerical or not. Furthermore decision tree algorithms can be implemented successfully even with missing data, although our dataset is complete and entirely numerical. The other reason is that the aggregated nature of our input space diminishes the impact of any outliers present.

Implementation

Our solution was implemented using python 2.7 as well as a few common python libraries for machine learning and data science. Pandas was used to load and handle the raw data, NumPy to handle the math efficiently, matplotlib for visualizations, and several modules of the scikit-learn library which include the “manifold” module to visualize the data as a scatterplot, the “model_selection” module to cross validate and split data into a training and testing set as well as to tune the hyper-parameters of the models, the “ensemble” module to actually implement the randomized forest and extremely randomized trees algorithms as well as the “tree” module for the simple tree benchmark, and the “metrics” module for the performance evaluation of the models.

Refinement

The results upon first implementing the rough version of this solution using randomized forests and extremely randomized forests are shown in the table below, along with our

benchmark. These F1 scores represent the average F1 score of a 3-fold cross validation on the dataset.

Initial results	
Model	F1 score
Decision Tree (benchmark)	0.9213
Random Forest	0.9599
Extremely Random Forest	0.9682

From these results, we see that ensemble learning does in fact represent a more desirable solution, but it is less clear which ensemble learning method is actually preferable, as the f1 scores are very close. Both warrant more investigation in the form of tuning their hyper-parameters. To tune these parameters, we use the "GridSearchCV" module of scikit-learn, which performs a cross validated exhaustive search over a specified grid of parameter values and identifies the models with the best performance. This specified parameter grid is detailed below.

Hyper-parameters tuned	
Hyper-parameter	Values represented in parameter grid
Number of estimators	10, 20, 30, 40 … 150
Maximum features to consider when splitting	5, 10, 15, 20, 25, 30
Minimum samples to split	2, 4, 6, 8

Upon the exhaustive search through this parameter grid with both random forests and extremely randomized forests using 3-fold cross validation of the training data, the best performing models found are detailed in the tables below

Best-performing Random Forest Hyper-parameters	
Hyper-parameter	Value
Number of estimators	130
Maximum features to consider when splitting	5
Minimum samples to split	2

Best-performing Extremely Random Forest Hyper-parameters	
Hyper-parameter	Value
Number of estimators	90
Maximum features to consider when splitting	20
Minimum samples to split	2

When the classifiers with these specific hyper-parameters were trained using 85% of the dataset and their performance scored, they produced the f1 scores shown below:

Training scores of fine-tuned models	
Model	F1 score on training data
Random Forest	1.000
Extremely Random Forest	1.000

Results

Model Evaluation and Validation

What we really want to know is how well this model generalizes to unseen data. To investigate this, their performance classifying the other 15% of the dataset which was held out during training was measured with the results shown below:

Testing scores of fine-tuned models	
Model	F1 score on testing data
Random Forest	0.9825
Extremely Random Forest	0.9913

Justification

Studying the performances of our tuned models, we see that they both represent improvements over our established benchmark (F1 = 0.9213) as well as their own untuned counterparts, with the tuned extremely random forests implementation performing a little better overall (F1 = 0.9913). All in all, our final solution represents an improvement over our benchmark with a difference in F1 scores of 0.07, which is a significant improvement. We can get an even better idea of our models performance by expressing its performance on the testing set using a confusion matrix, shown below:

Confusion matrix depicting Extremely Random Trees' performance on testing data		
	Predicted condition benign	Predicted condition malignant
True condition benign	57	1
True condition malignant	0	28

From this confusion matrix, we see that of our testing set which is 15% of 569 patients totaling 86 instances of data, 85 instances are classified correctly with only a single error. Furthermore, this single error is a type II error, or false negative, where the positive condition is a benign diagnosis. If there are to be errors in our models classification, we would prefer them to be type II rather than type I, or false positives, because in a real world situation it would be better to warn a patient they might be at risk of having breast cancer and to seek further medical attention than to tell a patient who might actually have breast cancer that they do not need further attention. For further context, the confusion matrix of our trained decision tree benchmark's testing performance, which was also relatively strong but with 300% the errors of our final tuned model, is shown below:

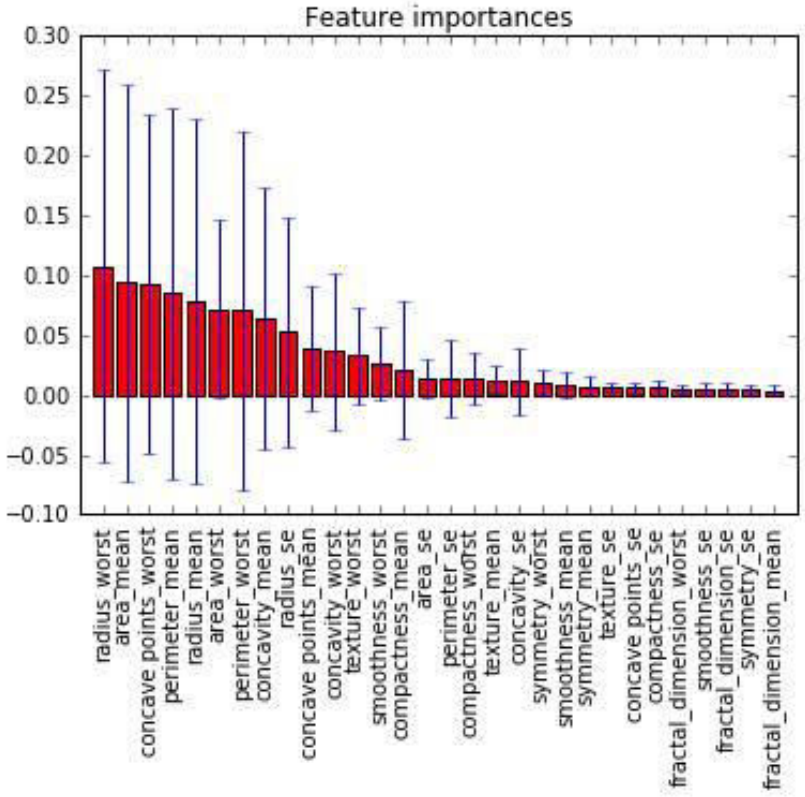
Confusion matrix depicting decision tree benchmark's performance on testing data		
	Predicted condition benign	Predicted condition malignant
True condition benign	55	3
True condition malignant	0	28

Our final version of this extremely random forest model aligns with our original expectations of what the solution to this classification task should be, generalizes well to unseen data, and overall, is a model which can be trusted help screen for breast cancer.

Conclusion

Free-Form Visualization

With an effective model for classifying this data now in hand, we can use its properties to investigate this problem and its solution to see if any insights can be gained. A good way to begin is to visualize the importance of each of the features in the input space. The more important features can be thought of as those which maximize the class purity on either side of the split were our algorithm to split on them. A visualization of the feature importances is shown below:



Earlier when we visualized the feature distributions for each diagnosis, we noted the contrast between the diagnoses for two feature distributions specifically, “concave points_worst” and “radius_worst”. From this feature importance visualization, we see that the features our algorithm identified as the most important and the third most important were “radius_worst” and “concave points_worst” with “area_mean”, “perimeter_mean”, and “radius_mean” (three directly correlated features) being ranked second fourth and fifth respectively, and “area_worst” and “perimeter_worst” (two features directly correlated to “radius_worst”) rounding out the top seven. Beyond these features there seem to be a handful of other features with significant importance and beyond those, 9-10 features with very little if any real relevance.

Reflection

In conclusion, in this project we have applied ensemble machine learning methods to a real world dataset in order to accurately distinguish cancerous breast tissue from non-cancerous breast tissue using only aggregated statistics of cell nuclei obtained from FNA samples, as well as learn what properties of the cell nuclei samples play a prominent role in this distinction. One challenge encountered in the implementation of this solution was encountered when manually creating the parameter grid which was to be exhaustively searched to deliver the best tuned model. With a number of potential hyper-parameters available to fine-tune, it was not exactly intuitive which ones to focus on and, subsequently, which values to include. Furthermore, the “GridSearchCV” module is rather computationally intensive for a large parameter grid, however, after some experimentation, a useful and succinct parameter grid was eventually settled upon.

Improvement

One way in which our solution could be improved and extended would be to take our final model and what we've learned about feature importances and combine this ensemble learning process with a computer vision module which could deliver an even larger and more detailed set of features obtained directly from the digital images of the FNA samples to an extremely randomized forest classifier. This solution could provide more accurate and precise classification because it would have a more detailed understanding of each FNA sample while still working with the broader understanding of learned feature importance.

Works Cited

[1] http://www.cancer.org/acs/groups/content/@research/documents/document/acspc-046381.pdf

[2] https://www.researchgate.net/figure/232811011_fig3_Captured-images-of-layers-of-glass-with-smears-of-breast-massobtained-by-FNA-the-parts

[3] https://www.kaggle.com/uciml/breast-cancer-wisconsin-data/downloads/breast-cancer-wisconsin-data.zip