Machine Learning Capstone Project Report

Project Overview
Breast cancer is the second leading cause of cancer death among women, only behind skin cancer. Every two minutes, a woman is diagnosed with breast cancer and every thirteen minutes a woman will die from it. It is estimated that 1 in 8 women in the U.S. will be diagnosed with breast cancer in their lifetime, this translates to 252,710 diagnoses every year in the U.S. alone [1].

Early screening and detection are crucial to survival. If breast cancer is detected early, there are more treatment options resulting in a 93% higher survival rate in the next five years [2]. The size of a breast cancer and how far it has spread are key factors in predicting the prognosis [2].

The Diagnostic Wisconsin Breast Cancer Database is a dataset compiled by members of the University of Wisconsin General Surgery and Computer Science departments that has been made publicly available since 1995 [3]. The dataset contains 569 samples of features extracted from digitized images of fine needle aspirate of breast mass, which is a type of biopsy in which a thin needle is inserted into an area of abnormal tissue or body fluid. Each sample contains 32 features that describe the data. The dataset contains 357 samples of benign tumors and 212 samples of malignant tumors so it is not an even distribution, almost 63% of the samples are benign.

The features describe various characteristics of the cell nuclei present in the image along with the final diagnosis, whether the cell is benign or malignant. The features include measurements such as radius, texture, perimeter, area, smoothness, compactness, symmetry, etc. These features are what doctors look for visually to determine whether a breast cancer cell is benign or malignant.

Problem Statement
Through medical imaging techniques such as mammograms, breast cancer can be detected visually by doctors before they can be felt in self-exams. However, there are not enough doctors to examine the results thoroughly due to the sheer number of examinations. With the shortage of doctors expected to last through 2030 [4], it becomes increasingly important for women to get access to fast and reliable screenings for breast cancer.

Machine learning techniques can be utilized to analyze images of breast mass obtained through various methods to detect the presence of tumors and more importantly, whether the tumor is benign or malignant. Benign tumors do not spread throughout the body and requires different treatment as opposed to malignant tumors, which is cancerous and has the ability to spread uncontrollably to surrounding issue [5]. This can be used to assist doctors by acting as a pre-screening check before they analyze the image themselves for confirmation, saving them time and acting as a sanity check. Doctors gain their expertise through years of training and experience, examining hundreds and thousands of images of breast cells to determine whether breast cancer exists based on the size, shape, smoothness, and other characteristics.

The solution I propose is to utilize machine learning techniques to train a model on the dataset that can reliably and accurately predict whether a breast cell is benign or malignant (binary classification) based on the physical characteristics of the cell such as size, shape, smoothness etc. The model can be used as an automated pre-check before a doctor examines the image themselves and act as a second opinion for them. Metrics such as accuracy, sensitivity, specificity, and F-score of the model can be used to measure the performance of the model.

The first step to any machine learning problem is to explore and try to understand the dataset to gain insight into any existing patterns. I will look at metrics such as mean, median, and mode for all the features, then look at them separated by class. I will try to see whether how the data is distributed across the features and classes by visually plotting them. It would also be a good idea to check for any missing data from the samples.

Next, I'll perform feature scaling for all of the features as part of the data pre-processing. Then, the dataset will be split into training and testing sets with either 70:30 train/test ratio, the same used by [6], or 80:20 train/test ratio. Since the dataset only contains 569 samples we may need more training data to prevent underfitting. I'll also explore using K-fold cross validation.

The dataset would be a good candidate for supervised learning techniques since the target variable (Diagnosis) is available. One of the supervised methods not covered by [6] is SVMs. I will train a SVM model on the dataset trying out different kernels and hyper-parameters and compare the performance. It would be good to evaluate the performance of the model when trained with all of the features versus a select set of features that are more important than the others, we can train the model on the top 1, 3, 5, and 10 features out of the total 32 features and see where the diminishing returns start to show. We can use the analysis from [6] to select the top important features.

For the SVM hyper-parameters gamma and C, I plan to do an exhaustive grid search to find the optimum values. A model will be trained using the optimum values of gamma and C found using grid search and the performance of the model will be measured through the evaluation metrics discussed in the previous section and compared with the results from [6].

I will then repeat the steps above for the AdaBoost model using decision trees, followed by a Random Forest classifier. Finally, the performance of the 3 models will be compared with each other as well as [6].

Metrics
For this specific problem, it is important that we aim to maximize the true positives and negatives while minimizing the false positives and negatives. In other words, we'd like to predict that only breast tumors that are actually malignant as malignant, and the same for breast tumors that are benign. This can be measured using the F1-score, which is the harmonic average of the precision and recall and given by the equation:

$$2 * \frac{precision * recall}{precision + recall}$$

Since the distribution of the 2 classes in the dataset is skewed towards one class, the F1-score will also be the main metric used to measure the performance of the solution.

The final prediction accuracy on the testing set will be used as a supporting metric to evaluate our model's performance. The prediction accuracy will show how often our model is correct in predicting whether a tumor is benign or malignant. The equation to calculate accuracy is given by the equation:

$$\text{Accuracy} = (TP+TN)/\text{Total}$$

Where TP=# of true positives and TN=# of true negatives.

Data Exploration

A sample of the dataset is shown below (first 10 columns):

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809

The diagnosis (target label) is designated by either a 'B' for benign or 'M' for malignant. The rest of the features are all positive float values. There are no missing values for any samples in the dataset.

The statistics for the dataset features are summarized below (first 9 columns):

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000

One notable item from the statistics is the standard deviation for the "area_mean" metric, which at 351 is very large given that the mean is 654 and the median is 551. This suggests that this metric may be very important for distinguishing between the two classes. Now I will break the stats down by class.
The statistics for all samples belonging to the malignant class is displayed below (first 9 columns):

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
count	212.0	212.000000	212.000000	212.000000	212.000000	212.000000	212.000000	212.000000	212.000000
mean	1.0	17.462830	21.604906	115.365377	978.376415	0.102898	0.145188	0.160775	0.087990
std	0.0	3.203971	3.779470	21.854653	367.937978	0.012608	0.053987	0.075019	0.034374
min	1.0	10.950000	10.380000	71.900000	361.600000	0.073710	0.046050	0.023980	0.020310
25%	1.0	15.075000	19.327500	98.745000	705.300000	0.094010	0.109600	0.109525	0.064620
50%	1.0	17.325000	21.460000	114.200000	932.000000	0.102200	0.132350	0.151350	0.086280
75%	1.0	19.590000	23.765000	129.925000	1203.750000	0.110925	0.172400	0.203050	0.103175
max	1.0	28.110000	39.280000	188.500000	2501.000000	0.144700	0.345400	0.426800	0.201200

The statistics for all samples belonging to the benign class is displayed below (first 9 columns):

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
count	357.0	357.000000	357.000000	357.000000	357.000000	357.000000	357.000000	357.000000	357.000000
mean	0.0	12.146524	17.914762	78.075406	462.790196	0.092478	0.080085	0.046058	0.025717
std	0.0	1.780512	3.995125	11.807438	134.287118	0.013446	0.033750	0.043442	0.015909
min	0.0	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000
25%	0.0	11.080000	15.150000	70.870000	378.200000	0.083060	0.055620	0.020310	0.015020
50%	0.0	12.200000	17.390000	78.180000	458.400000	0.090760	0.075290	0.037090	0.023440
75%	0.0	13.370000	19.760000	86.100000	551.100000	0.100700	0.097550	0.059990	0.032510
max	0.0	17.850000	33.810000	114.600000	992.100000	0.163400	0.223900	0.410800	0.085340

It is apparent that there are notable differences between samples from the two classes based on their statistics. For all of the features, it can be seen that the mean values for malignant samples are noticeably higher than those for benign samples. We can get a better picture by calculating the difference as a percentage:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
count	40.616246	40.616246	40.616246	40.616246	40.616246	40.616246	40.616246	40.616246	40.616246
mean	inf	43.768130	20.598341	47.761482	111.408198	11.268500	81.292962	249.072995	242.141814
std	NaN	79.946652	5.397946	85.092262	173.993502	6.231125	59.963164	72.687876	116.068815
min	inf	56.854319	6.900103	64.192738	151.986063	40.053202	137.616099	inf	inf
25%	inf	36.055957	27.574257	39.332581	86.488630	13.183241	97.051420	439.266371	330.226365
50%	inf	42.008197	23.404255	46.073164	103.315881	12.604672	75.786957	308.061472	268.088737
75%	inf	46.522064	20.268219	50.900116	118.426783	10.153923	76.729882	238.473079	217.363888
max	inf	57.478992	16.178645	64.485166	152.091523	11.444308	54.265297	3.894839	135.762831

In the table above, the values represent the percentage difference between statistics of malignant samples versus benign samples. For example, the mean values for “radius_mean” of malignant samples are 43.7% higher compared to the mean values for “radius_mean” of benign samples. In the features displayed, the only feature that did not show a significant change is “smoothness_mean”, which exhibited only a 11% difference. We can then sort the percentage difference table to see which features had the biggest difference between the two classes and which features showed the smallest difference. Here’s the table showing the first 9 features that show the greatest percentage difference between the two classes sorted by the mean of the values:

	diagnosis	concavity_mean	area_se	concave points_mean	concavity_worst	area_worst	concave points_worst	perimeter_se	radius_se	area_mean
count	40.616246	40.616246	40.616246	40.616246	40.616246	40.616246	40.616246	40.616246	40.616246	40.616246
mean	inf	249.072995	243.846204	242.141814	171.060959	154.479826	144.796717	116.161737	114.403514	111.408198
std	NaN	72.687876	593.791625	116.068815	29.308003	265.502774	29.360859	233.071607	206.511310	173.993502
min	inf	inf	105.674802	inf	inf	174.352052	inf	76.221929	73.811659	151.986063
25%	inf	439.266371	134.354522	330.226365	323.488583	117.020801	199.275078	87.923875	88.314038	86.488630
50%	inf	308.061472	197.784004	268.088737	186.756374	138.034344	144.919930	98.784441	112.504854	103.315881
75%	inf	238.473079	275.549341	217.363888	150.981498	155.634328	116.099087	118.017169	121.692037	118.426783
max	inf	3.894839	603.151342	135.762831	6.549521	251.570248	66.285714	329.464635	226.069686	152.091523

Here's the table sorted by the median of the values:

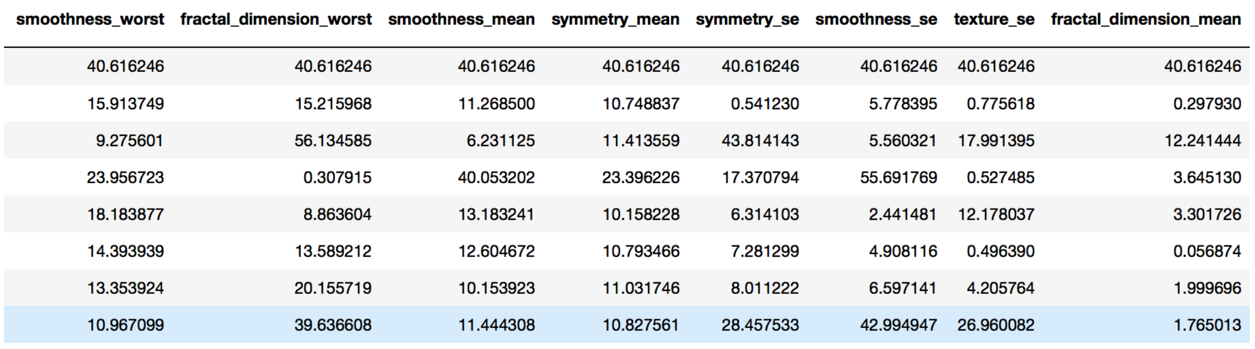
	diagnosis	concavity_mean	concave points_mean	area_se	concavity_worst	concave points_worst	area_worst	radius_se	compactness_worst	area_mean
count	40.616246	40.616246	40.616246	40.616246	40.616246	40.616246	40.616246	40.616246	40.616246	40.616246
mean	inf	249.072995	242.141814	243.846204	171.060959	144.796717	154.479826	114.403514	105.189070	111.408198
std	NaN	72.687876	116.068815	593.791625	29.308003	29.360859	265.502774	206.511310	84.825366	173.993502
min	inf	inf	inf	105.674802	inf	inf	174.352052	73.811659	88.017589	151.986063
25%	inf	439.266371	330.226365	134.354522	323.488583	199.275078	117.020801	88.314038	118.281250	86.488630
50%	inf	308.061472	268.088737	197.784004	186.756374	144.919930	138.034344	112.504854	109.864547	103.315881
75%	inf	238.473079	217.363888	275.549341	150.981498	116.099087	155.634328	121.692037	94.548219	118.426783
max	inf	3.894839	135.762831	603.151342	6.549521	66.285714	251.570248	226.069686	80.885621	152.091523

From the above 2 tables it can be seen that the top features that exhibit the largest difference between the 2 classes are almost identical whether sorted by the mean or media. Features such as "concavity_mean", "area_se", and "concave_points_mean" show over 200% difference between the two classes, suggesting these values should be important to use in our training.

Now we can look at the same tables for the features with the smallest percentage difference between the two classes. First look at features sorted by mean:

fractal_dimension_worst	fractal_dimension_se	smoothness_mean	symmetry_mean	smoothness_se	texture_se	symmetry_se	fractal_dimension_mean
40.616246	40.616246	40.616246	40.616246	40.616246	40.616246	40.616246	40.616246
15.215968	11.725753	11.268500	10.748837	5.778395	0.775618	0.541230	0.297930
56.134585	30.519203	6.231125	11.413559	5.560321	17.991395	43.814143	12.241444
0.307915	21.479660	40.053202	23.396226	55.691769	0.527485	17.370794	3.645130
8.863604	29.616683	13.183241	10.158228	2.441481	12.178037	6.314103	3.301726
13.589212	33.173077	12.604672	10.793466	4.908116	0.496390	7.281299	0.056874
20.155719	17.207714	10.153923	11.031746	6.597141	4.205764	8.011222	1.999696
39.636608	56.970509	11.444308	10.827561	42.994947	26.960082	28.457533	1.765013

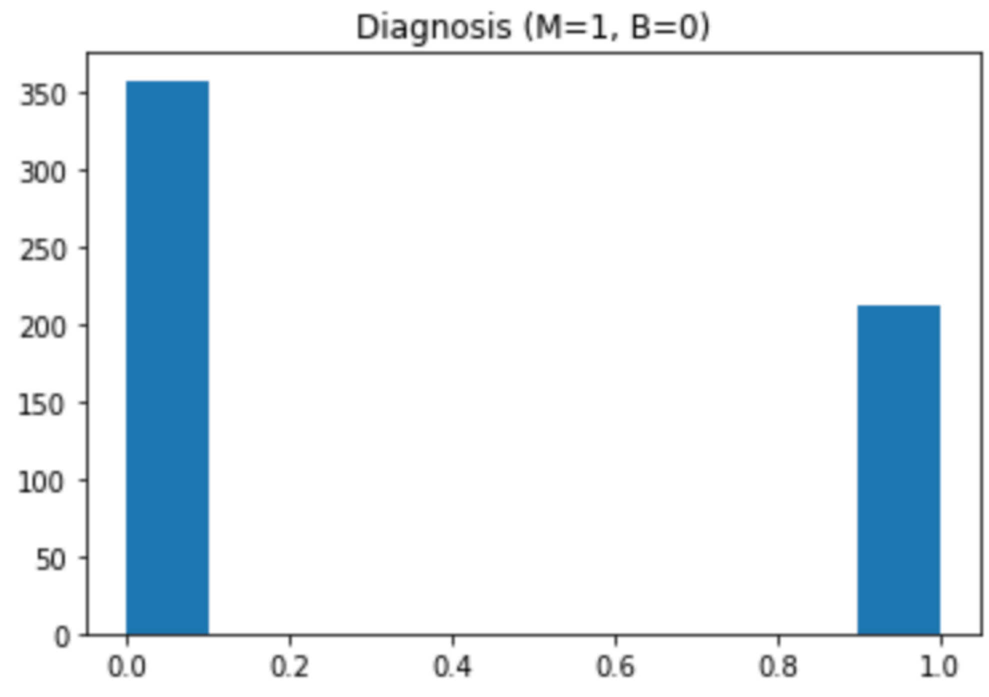
Now by the median:



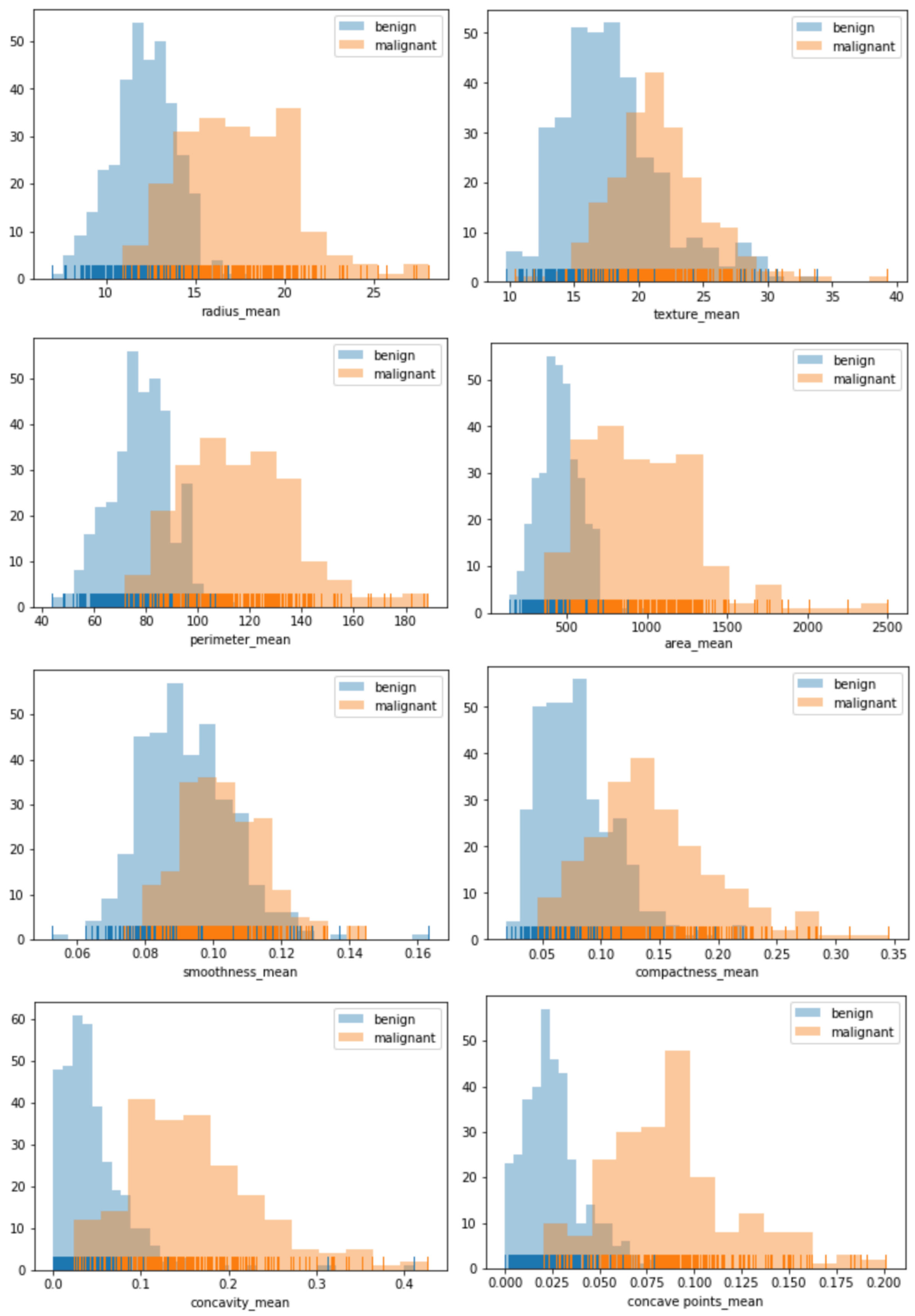
smoothness_worst	fractal_dimension_worst	smoothness_mean	symmetry_mean	symmetry_se	smoothness_se	texture_se	fractal_dimension_mean
40.616246	40.616246	40.616246	40.616246	40.616246	40.616246	40.616246	40.616246
15.913749	15.215968	11.268500	10.748837	0.541230	5.778395	0.775618	0.297930
9.275601	56.134585	6.231125	11.413559	43.814143	5.560321	17.991395	12.241444
23.956723	0.307915	40.053202	23.396226	17.370794	55.691769	0.527485	3.645130
18.183877	8.863604	13.183241	10.158228	6.314103	2.441481	12.178037	3.301726
14.393939	13.589212	12.604672	10.793466	7.281299	4.908116	0.496390	0.056874
13.353924	20.155719	10.153923	11.031746	8.011222	6.597141	4.205764	1.999696
10.967099	39.636608	11.444308	10.827561	28.457533	42.994947	26.960082	1.765013

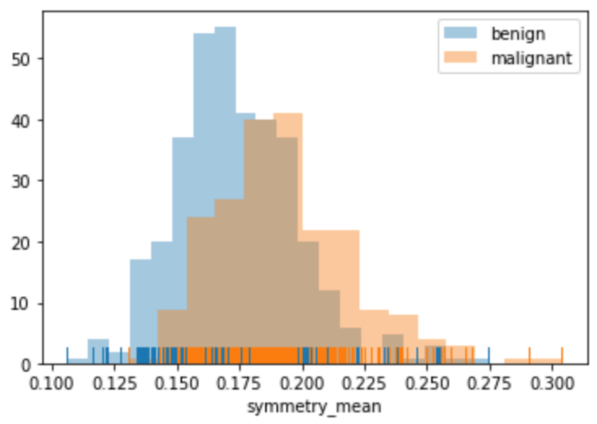
Again, the features that show the smallest difference between the two classes are almost identical whether sorted by mean or median. Features like "texture_se", "fractal_dimension_mean", and "symmetry_se" show less than 1% difference between the two classes, which suggests these features would not be useful is distinguishing between the two classes.

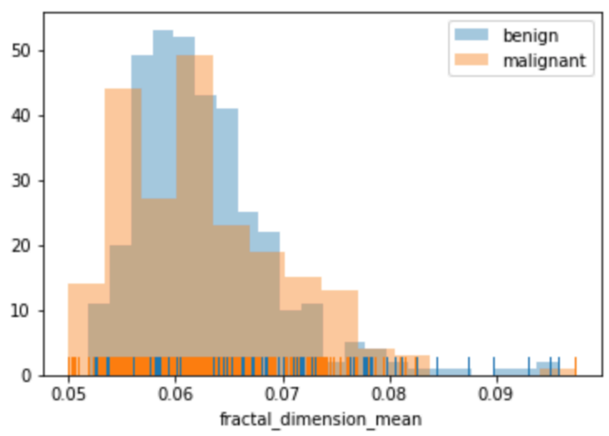
Out of a total of 569 samples in the dataset, 357 are labeled as benign and 212 as malignant. This gives us an uneven dataset skewed towards benign samples. This can be seen in the chart below:

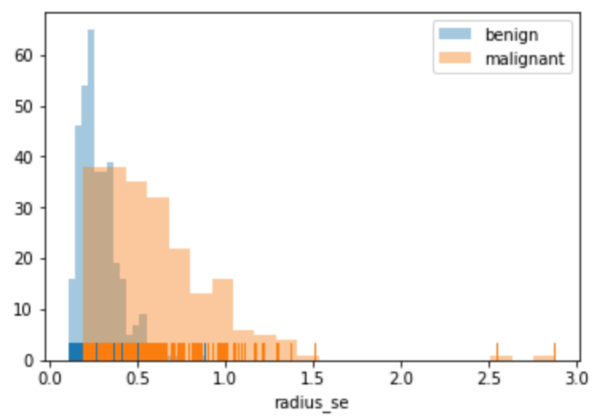


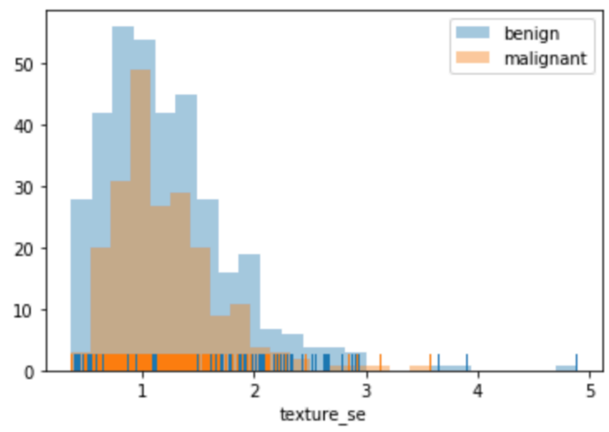
Exploratory Visualization
Another way to explore the dataset is to visually plot the distribution of the dataset's features for both classes.

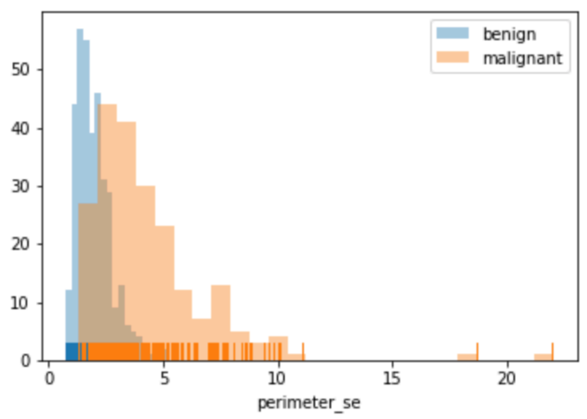

benign
malignant
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave points_mean

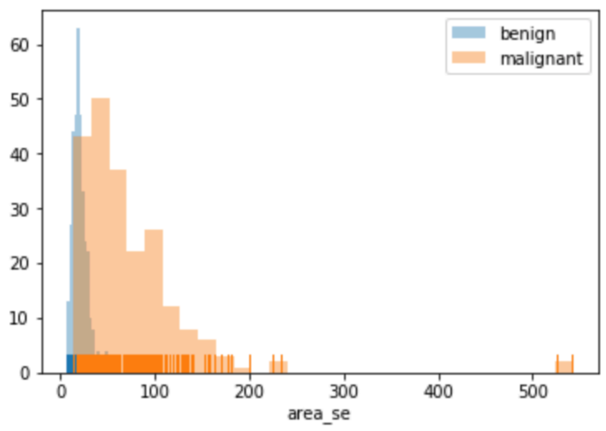

benign
malignant
symmetry_mean

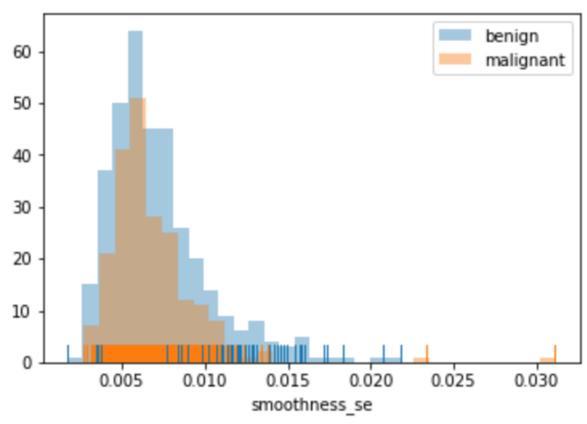

benign
malignant
fractal_dimension_mean

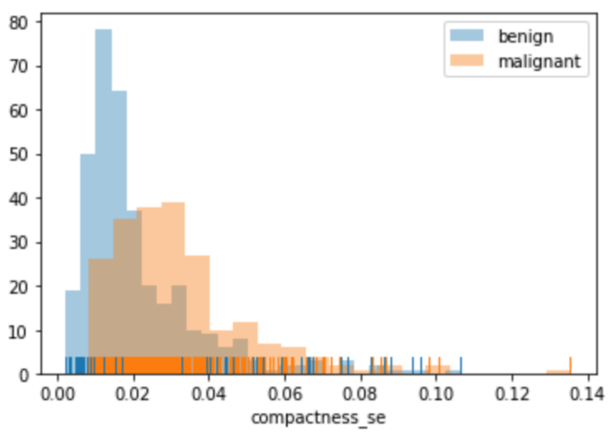

benign
malignant
radius_se

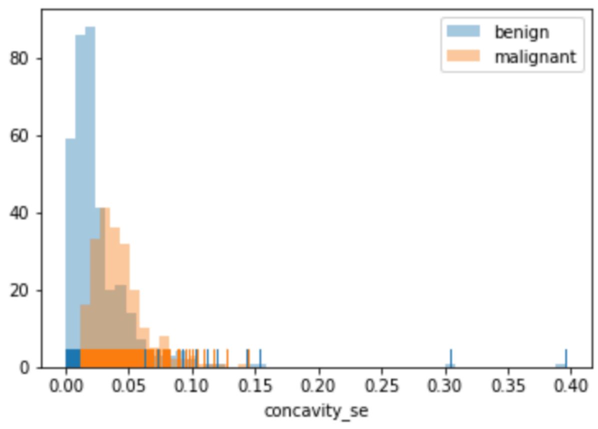

benign
malignant
texture_se

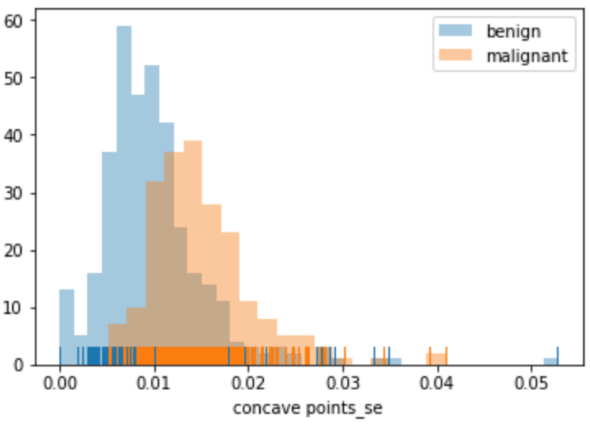

benign
malignant
perimeter_se

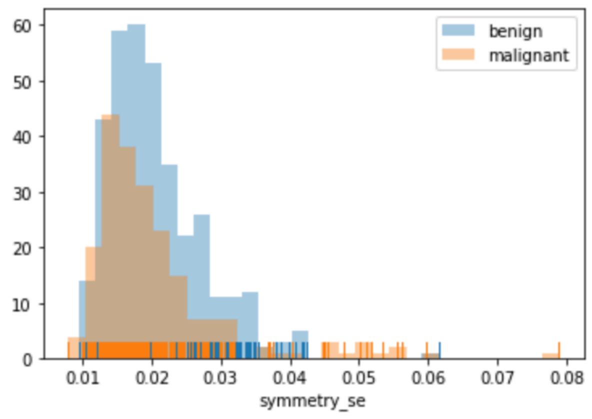

benign
malignant
area_se

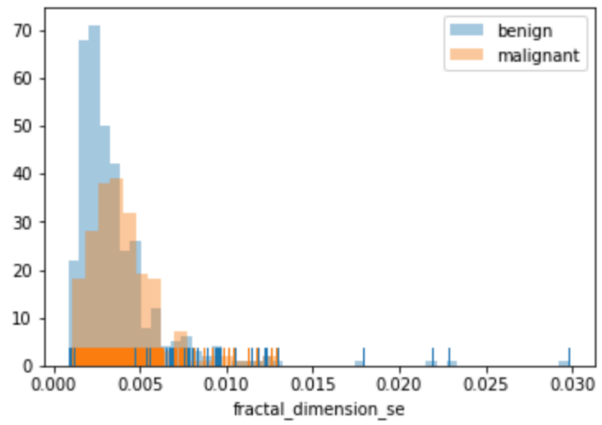

benign
malignant
smoothness_se

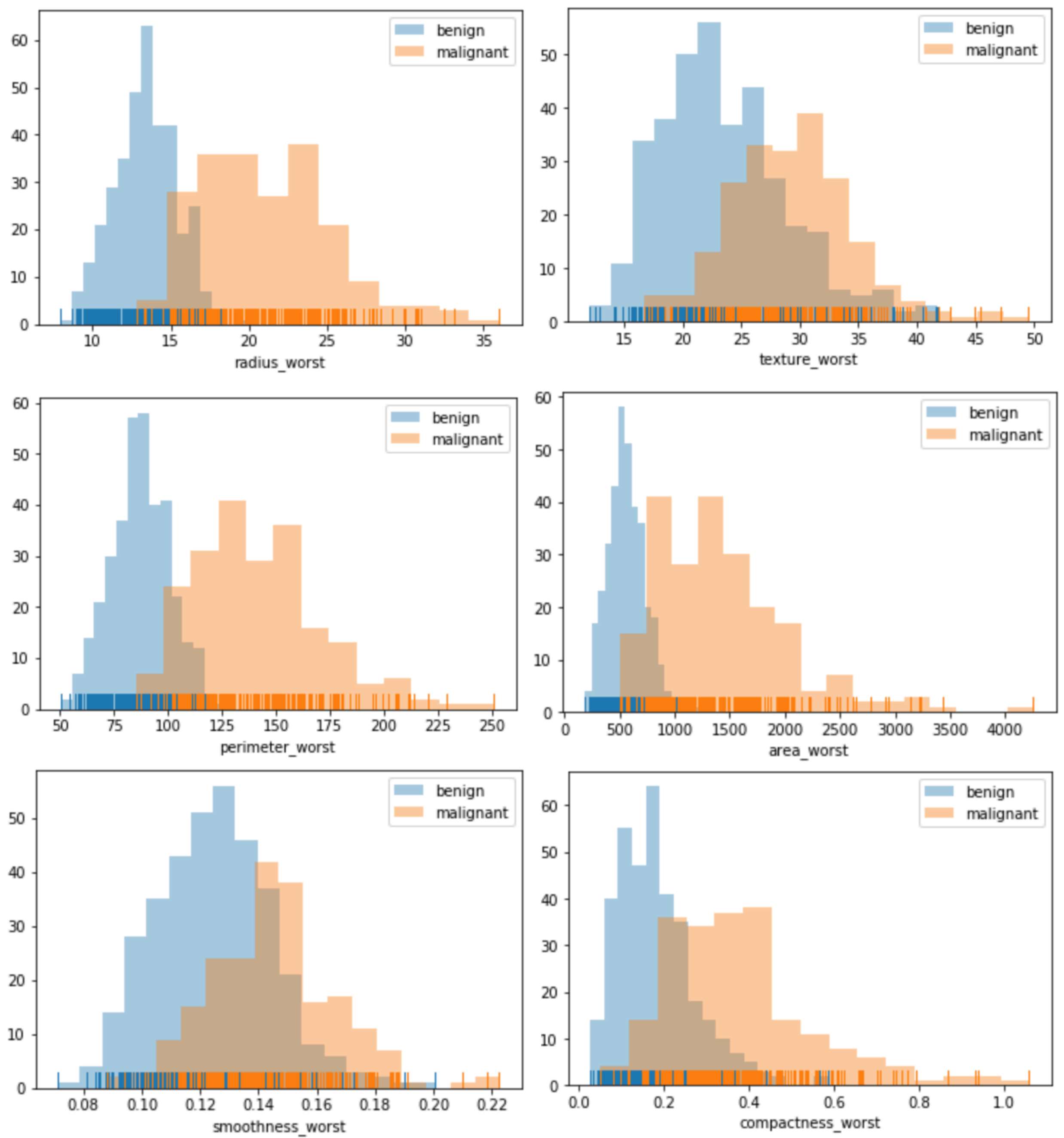

benign
malignant
compactness_se

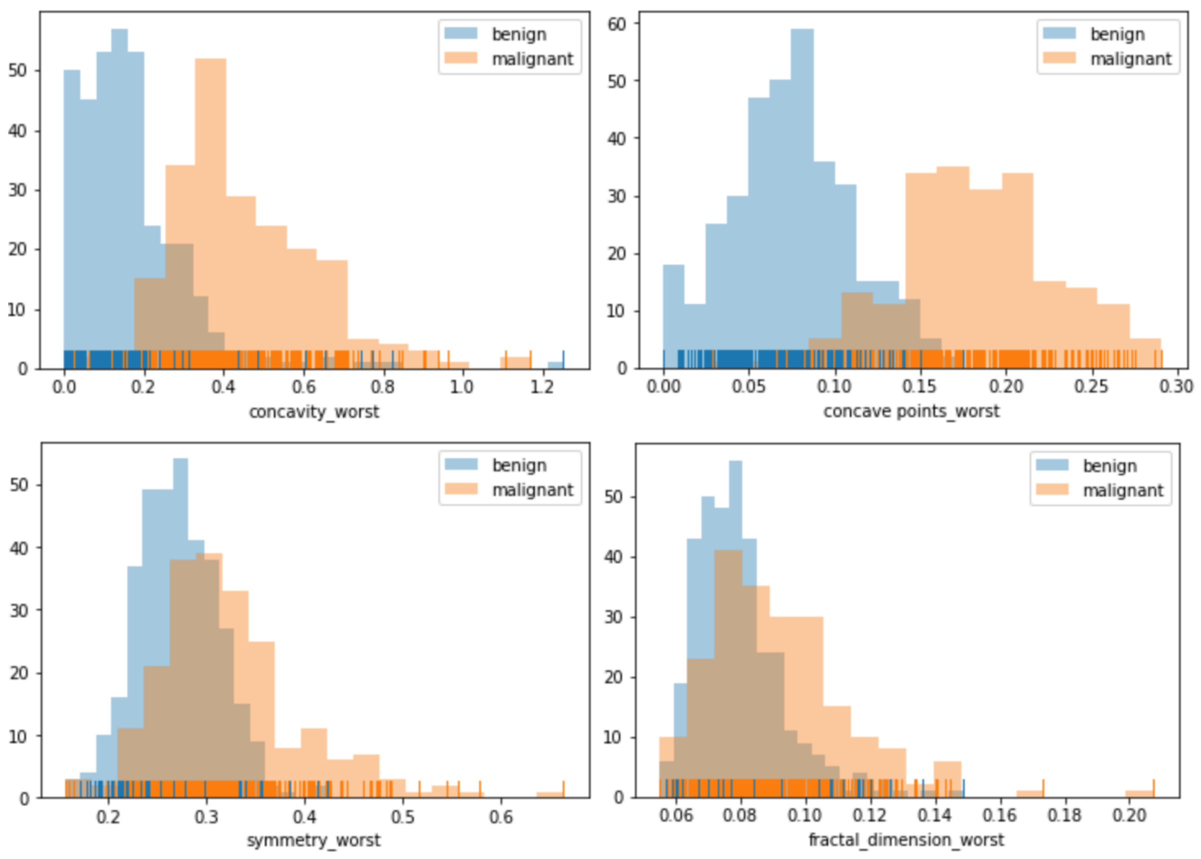

benign
malignant
concavity_se


benign
malignant
concave points_se


benign
malignant
symmetry_se


benign
malignant
fractal_dimension_se


benign
malignant
radius_worst
texture_worst
perimeter_worst
area_worst
smoothness_worst
compactness_worst



From the distribution plots, it can be seen that there are some features that have a lot of overlap between the two classes such as symmetry_worst and fractal_dimension_worst, these features should in theory not be useful in classication algorithms since both classes share a lot of the same values. On the other hand, the charts for features concave_points_mean and concave_points_worst show a clear separation between the two classes with minimal overlap. These features should provide more discriminatory power for training algorithms for classification.

Any outliers in the data can be easily picked up from the distribution plots. For example, concavity_worst, symmetry_worst, and fractal_dimension_worst all have a couple of malignant samples that are clear outliers on the right side of the charts. Concavity_worst also shows a benign outlier that has a higher value than even the malignant samples.

Algorithms and Techniques

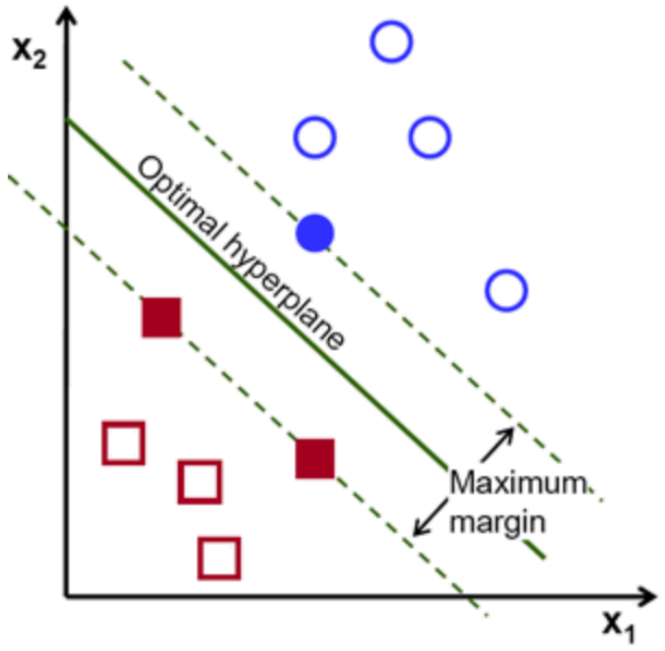
The dataset will be split into two subsets, one used for training and the other for testing. During training, I will employ K-fold cross validation on the training set using 3 stratified folds.

For this classification problem, I will explore 3 different supervised machine learning algorithms:

1. Support Vector Machines
2. Adaboost
3. Random Forest

Since this is a classification problem and the target label is given by the dataset, supervised learning algorithms are a good candidate.

A support vector machine works by creating a hyperplane that best separates two classes of data in a feature space. This is done by maximizing the margin between samples from different classes. An example of such a hyperplane can be shown in the example below [7]:



AdaBoost is a boosting algorithm developed for binary classification that creates a strong classifier from a number of weak classifiers. This is done by first building a model from the training data, and then building subsequent models that each attempt to correct the errors made the by previous model until either the training set is predicted perfectly or the maximum number of models is reached. AdaBoost is commonly used to boost the performance of weak learners such as decision trees. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction [8].

A random forest is another ensemble method that aims to improve the performance of decision trees by introducing randomness. It works by fitting a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy of the model. In random forests, each tree in the ensemble is built from a sample drawn with replacement from the training set. When splitting a node in the tree, the split that is picked is the best split among a random subset of the features instead of the best split among all features. This randomness usually results in a higher variance and an overall better model [9].

For all 3 models, I will start initially with the default parameters assigned by sklearn. For selecting the features to use for each algorithm I will do the following:

1. Train using all features
2. Train using top 10 features according by [6]

3. Train using top 5 features according by [6]
4. Train using top 3 features according by [6]
5. Train using top 1 feature according by [6]

I will then compare the performance and select the features that yield the best results and use that to perform a grid search to find the optimum parameters and hyperparameters for each respective model. For SVM this will be the gamma and C; n_estimators, learning_rate, and algorithm for AdaBoost; n_estimator, max_depth, and min_samples_split for Random Forest.

Once the best parameters for each algorithm have been found using grid search, I will train a final model using those parameters and the selected feature set and evaluate its performance and see if parameter optimization helps performance.

For the Random Forest model, after training it on all the features I will check the feature rankings provided by the model to see which features were most significant during training. Then I will select the top 10, 5, 3, and 1 features from that set and repeat the process using the top features determined by my Random Forest model as opposed to those determined by [6].

Benchmark
The benchmark that I will use to compare my results across the 3 models with are those determined by [6]. [6] found that the best model was the Random Forest classifier using the top 5 features which achieved an accuracy of 96.49% on the testing set. Since they only used accuracy as the performance metric I cannot compare my model's F1 score directly with their results. However, for this problem where false positives and false negatives can lead to bad things we should aim for an F1 score greater than 0.9.

Data Preprocessing
For preprocessing I remapped the values for the diagnosis output from characters to integers. Malignant samples labeled as 'M' were mapped to 1, and benign samples labeled as 'B' were mapped to 0. The values for all the features were also scaled to the range (0, 1), this was needed because some features had a large range and also outliers that would skew the data when used in the training algorithms without scaling.

Implementation
I used various python libraries such as numpy, pandas, seaborn, matplotlib, and sklearn for the implementation of the solution so most of the algorithms were already implemented.

First I imported the dataset into the python environment using pandas.read_csv(), then the target label values had to be remapped from characters to integers using the pandas.Series.map() function. Statistics about the dataset were obtained using the pandas.dataframe.describe() function. Then the distribution plots for the features were done using seaborn.distplot().

A function "scale_data" was created that takes input data and performs scaling using sklearn's MinMaxScaler() and returns the scaled data. All features used to train the models are scaled using this function.

```python
# Scale the data using MinMaxScaler and return transformed data
def scale_data(data):
    scaler = MinMaxScaler()
    scaler.fit(data)
    return scaler.transform(data)
```

A function was created that takes as input the dataset, features, targets, and parameters relevant for SVC, and creates a sklearn.SVC model using those parameters and trains the model on the selected features and targets from the dataset. The trained model is returned for later use. All the SVM models were trained using this function on the training dataset and the resulting trained model is used to predict on the testing set.

```python
# Train a SVC model on the selected data with the given params and return fitted model
def train_svc_model(data, features, targets, kern='rbf', gamm='auto', c=1.0, deg=3):
    clf = SVC(kernel=kern, gamma=gamm, C=c, degree=deg)

    # Get features and targets to use
    X = data[features]
    y = data[targets]

    # Scale data
    X = scale_data(X)
    y = scale_data(y)

    # Do stratified K-fold CV with 3 splits
    kf = StratifiedKFold(n_splits=3)
    for k, (train, test) in enumerate(kf.split(X, y)):
        train_X = (data[features].iloc[train,:])
        train_y = data[targets].iloc[train]
        test_X = (data[features].iloc[test,:])
        test_y = data[targets].iloc[test]

        # Fit model on training subset
        clf.fit(train_X, train_y)
        print("[fold {0}], score: {1:.5f}".
          format(k, clf.score(test_X, test_y)))

    # Fit model on data
    clf.fit(X, y)
    print("Training accuracy: %s" % "{0:.3%}".format(clf.score(X, y)))

    return clf
```

Similar to the SVC training function, functions were also created to perform training for the AdaBoost and Random Forest classifiers. The fitted models returned by these training functions are then passed to a prediction function that predicts the outcome and calculates the performance metrics.

The prediction function takes as input a model, dataset, features, and targets, the input model’s predict() function is used on the dataset’s features and targets to make predictions. The sklearn.metrics.f1_score and sklearn.metrics.accuracy are used to get the performance metrics.

```
# Predict the target labels on the data using the model and print F1 score and accuracy
def test_svc_model(model, data, features, targets):
    # Get features and targets to use
    X = data[features]
    y = data[targets]

    # Scale data
    X = scale_data(X)
    y = scale_data(y)

    # Predict target labels using the features
    predictions = model.predict(X)

    # Calculate accuracy and F1 score
    accuracy = metrics.accuracy_score(predictions, y)
    f1_score = metrics.f1_score(y, predictions)

    print("F1 score: %s" % "{0:.3}".format(f1_score))
    print("Accuracy: %s" % "{0:.3%}".format(accuracy))
```

An example workflow for training a SVM model using all the features is shown below:

```
# Train SVC using all features on training set
trained_model = train_svc_model(data_train, list(data_features), 'diagnosis')
```

```
# Test trained SVC model using all features on testing set
test_svc_model(trained_model, data_test, list(data_features), 'diagnosis')
```

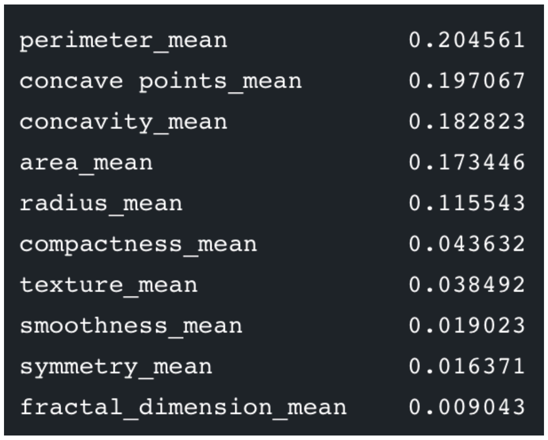
For the grid search I created arrays to hold the different values for the parameters I wanted to search over and used the sklearn GridSearchCV to do the grid search. Here’s the grid search implementation I used for the SVM mode:

```
# Do grid search for best C and gamma for SVC using the top 3 features
C_range = np.logspace(0, 7, 8)
gamma_range = np.logspace(-4, 3, 8)
print C_range
print gamma_range

parameters = {'kernel':('rbf', 'sigmoid'), 'C':C_range, 'gamma':gamma_range}
svc = SVC()
clf = GridSearchCV(svc, parameters, n_jobs=1, cv=3, verbose=1)

X = data_train[selected_features_top3]
y = data_train['diagnosis']
X = scale_data(X)
y = scale_data(y)

clf.fit(X, y)
print clf.best_score_
print clf.best_params_
```

The grid search implementation for the AdaBoost and Random Forest models are similar.

There were no major complications during the coding process, the hardest part was validating that the preprocessing steps such as remapping target labels and scaling the values were done correctly by printing the values before and after.

Refinement
After training the 3 models using the default parameter values assigned by sklearn for the different feature subsets, I looked at the feature subset that performed the best in terms of F1 score and accuracy and used that feature subset to perform a grid search for the optimal parameters for that specific model. As a reminder, here the top features were those derived by [6], which consisted of the following features along with their weights:



```
perimeter_mean            0.204561
concave points_mean       0.197067
concavity_mean            0.182823
area_mean                 0.173446
radius_mean               0.115543
compactness_mean          0.043632
texture_mean              0.038492
smoothness_mean           0.019023
symmetry_mean             0.016371
fractal_dimension_mean    0.009043
```

For the SVM model the initial default parameters were:

Kernel	RBF
Gamma	Auto (1/number of features)
C	1.0

The performance of the various feature subsets on the test set trained on the training set using the above params are summarized below:

Features	**F1 score**	**Accuracy (%)**
All features	0.884	90.351
Top 10	0.921	93.86
Top 5	0.907	92.982
Top 3	0.918	93.86
Top 1	0.871	90.351

I selected the top 3 features to use for grid search over the top 10 features in this case because they have pretty much identical performance with respect to both F1 score and accuracy, but 3 features is a lot less than 10 and we can try to avoid the curse of dimensionality.

After grid search the best parameters were:

Kernel	RBF
Gamma	10.0
C	10.0

The results after training a SVM classifier using the parameters above yields an accuracy of 93.86% and a F1 score of 0.918, which is identical to the results using the default hyperparameters. Since the hyperparameters found using grid search did not differ greatly from the default this result seems reasonable.

For the Adaboost model the default parameters were:

N_estimators	50
Learning rate	1.0
Algorithm	SAMME.R

The performance of the various feature subsets on the test set trained on the training set using the above params are summarized below:

Features	**F1 score**	**Accuracy (%)**
All features	0.707	70.175
Top 10	0.839	86.842
Top 5	0.828	86.842
Top 3	0.851	88.596
Top 1	0.747	81.579

I again selected the top 3 features to use for the grid search as it achieved the best performance. After grid search the best parameters were:

N_estimators	50
Learning rate	1.0e-08
Algorithm	SAMME

The results after training an AdaBoost classifier using the parameters above yields an accuracy of 91.228% and a F1 score of 0.881. This is an improvement over the default parameters.

For the Random Forest model the default parameters were:

N_estimators	10
Max_depth	Auto (sqrt(n_features)
Min_samples_split	2

The performance of the various feature subsets on the test set trained on the training set using the above params are summarized below:

Features	**F1 score**	**Accuracy**
All features	0.863	88.596
Top 10	0.929	94.737
Top 5	0.92	93.86
Top 3	0.782	83.333
Top 1	0.849	90.351

I selected the top 10 features to use for the grid search since it has the best performance.

After grid search the best parameters were:

N_estimators	10
Max_depth	6
Min_samples_split	10

The results after training a Random Forest classifier using the parameters above yields an accuracy of 95.614% and a F1 score of 0.941. This is an improvement over the default parameters.

After training a Random Forest classifier using all the features, we can get a feature importance list from the model along with their weights designating how important each feature is. The results are summarized in the table below:

perimeter_worst	0.246738
area_mean	0.187863
perimeter_mean	0.065499
perimeter_se	0.060154
radius_se	0.055068
compactness_mean	0.054497
radius_worst	0.047963
concavity_mean	0.039773
concave points_mean	0.037824
concave points_worst	0.036899
concavity_worst	0.024982
area_worst	0.024925
texture_worst	0.021528
symmetry_worst	0.016126
symmetry_mean	0.013853
texture_mean	0.013517
area_se	0.010363
compactness_worst	0.006394
fractal_dimension_worst	0.006210
fractal_dimension_se	0.005708
radius_mean	0.004567
symmetry_se	0.004543
texture_se	0.003673
smoothness_worst	0.003395
smoothness_mean	0.002250
smoothness_se	0.001671
fractal_dimension_mean	0.001625
concavity_se	0.001182
compactness_se	0.001104
concave points_se	0.000107

There is a significant drop off in importance after the first two features, and the features at the bottom of the list contribute almost nothing to the model.

The top 10 features extracted from this feature importance list in order of importance are: perimeter_worst, area_mean, perimeter_mean, perimeter_se, radius_se, compactness_mean, radius_worst, concavity_mean, concave points_mean, and concave points_worst. This list differs from [6] because they did not use all features for their Random Forest model.

Now I will perform the evaluations again using the top X features I obtained in place of the ones found by [6].

The performance of the various feature subsets on the test set trained on the training set using the default SVM params are summarized below:

Features	**F1 score**	**Accuracy (%)**
Top 10	0.95	96.491
Top 5	0.895	92.982
Top 3	0.833	89.474
Top 1	0.765	85.965

The best feature set to use for grid search is top 10 features.

After grid search the best parameters were:

Kernel	RBF
Gamma	0.01
C	100000

The results after training a SVM classifier using the parameters above yields an accuracy of 95.614% and a F1 score of 0.938, which is slightly lower than using the default parameters. This is most likely due to overfitting on the training set since the C is so large.

The performance of the various feature subsets on the test set trained on the training set using the default AdaBoost params are summarized below:

Features	**F1 score**	**Accuracy (%)**
Top 10	0.897	92.105
Top 5	0.854	87.719
Top 3	0.85	89.474
Top 1	0.813	85.088

I will select the top 10 features to use in grid search.

After grid search the best parameters were:

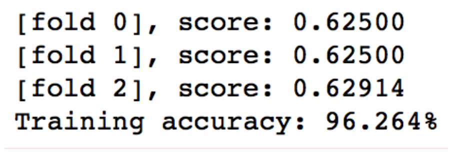
N_estimators	400
Learning rate	1.0e-08
Algorithm	SAMME

The results after training an AdaBoost classifier using the parameters above yields an accuracy of 92.105% and a F1 score of 0.892. This is about the same performance as using the default parameters.

Model Evaluation and Validation

The best performing model was the SVM model using the default parameters trained using the top 10 features found by our Random Forest model. This model achieved a prediction accuracy of 96.491% and F1 score of 0.95 on the testing set that was not used in training or grid search. This is a reasonable result given what we know about the features and the size of the dataset.

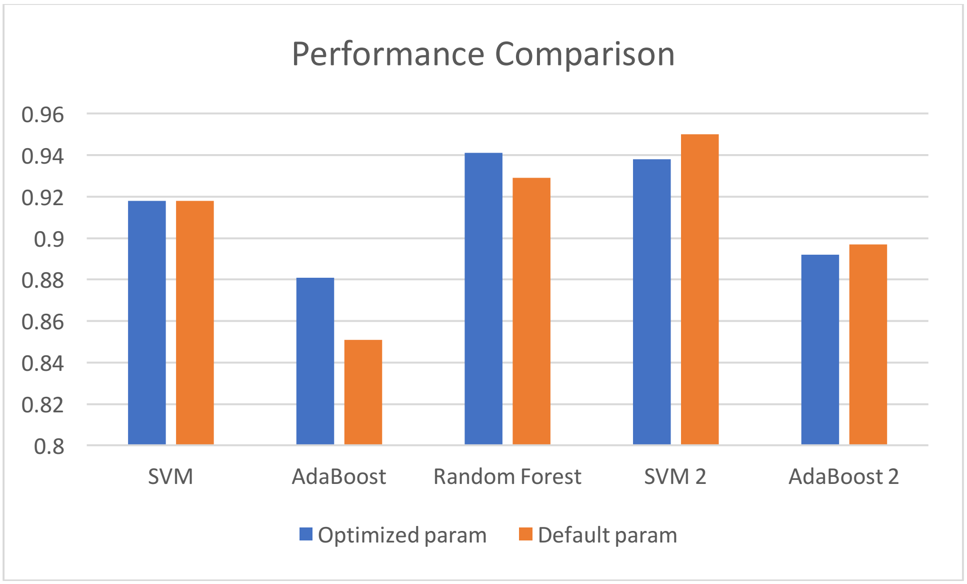
Even though we got good results from this model, I would not say this is a robust model for this problem because the dataset was too small to encompass all possibilities. With a dataset of 569 samples containing 357 benign and 212 malignant samples, after doing an 80:20 split into training and testing sets, there are even less data available for training and testing. I performed a StratifiedKFold Cross Validation on my final model with K=3 and got poor results:



```
[fold 0], score: 0.62500
[fold 1], score: 0.62500
[fold 2], score: 0.62914
Training accuracy: 96.264%
```

Justification
The best accuracy achieved by [6] was 96.491% using their top 5 features trained on the Random Forest model. My best accuracy was also 96.491% using my top 10 features trained on the SVM model. My model also achieved a F1 score of 0.95, which is a very respectable result. Since [6] did not report their F1 score I cannot compare the two models directly. However just based on the accuracy I would say my model is at least on par with theirs.

Free-form Visualization
Here is a chart to visually summarize the performance of the models I explored. The chart plots the F1 score of each model's best performance using both the default parameters and optimum parameters found using grid search side by side. Note that SVM 2 and AdaBoost 2 represent the respective models trained using the top features found by our Random Forest model instead of those found by [6].



It is interesting to note that if using top features found by [6], the performance improves using optimized parameters over the default parameters. On the other hand, when the top features found using our Random Forest model is used, the performance is slightly worse using the optimized parameters.

The AdaBoost model seem to have the worst performance while the SVM and Random Forest models have the best performance.

Reflection
Overall this was a very interesting problem that was fun to work on. Not only is this a real-life problem that many women will have to deal with, but for which any solution can have great impact on the future of healthcare for cancer diagnosis. I approached the problem by first analyzing the raw dataset and obtaining basic stats such as mean and median for the features, then getting an idea for the type of problem, which in this case is a classification problem. Next, I understood what kind of dataset I was working with respect to the number of samples and the distribution of the classes.

The models I explored using were the SVM, AdaBoost, and Random Forest. I took an exhaustive approach by training each model using different feature subsets to determine the optimum number of features, and then used that to do a grid search for the optimum parameters for each model. Through this repetitive process of optimizing the model inputs and parameters, we were able to continuously improve our results.

One aspect of the project that I found particular interesting was the result of the StratifiedKFold cross validation on my final model. Even though it achieved a training and testing accuracy of over 96% and a F1 score of 0.95 on the testing set, the scores for the 3 folds were around 0.62 on the training set.

The final model and its performance is very good and met my expectations. However, due to reasons I outlined in an earlier section with respect to the small dataset size, I would not use this solution in a general setting for this problem today in the field.

Improvement
I think one improvement I can make to my solution is to combine the 3 different types of models I used into a final voting classifier that will allow the models to vote on the output target label. This way instead of saying that a sample is predicted to be malignant or benign, we have the option of saying that a sample is predicted to be malignant or benign with a certain confidence level depending on how the 3 models voted in the end.

Works Cited

[1] National Breast Cancer Foundation, "Breast Cancer Facts," National Breast Cancer Foundation, 2016. [Online]. Available: http://www.nationalbreastcancer.org/breast-cancer-facts. [Accessed 29 11 2017].

[2] Carol Milgard Breast Center, "Early Detection is Key," 2017. [Online]. Available: http://www.carolmilgardbreastcenter.org/early-detection. [Accessed 30 11 2017].

[3] UC Irvine Machine Learning Repository, "Breast Cancer Wisconsin (Diagnostic) Data Set," 2007. [Online]. Available:

https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29. [Accessed 1 12 2017].

[4] S. Mann, "New Research Shows Shortage of More than 100,000 Doctors by 2030," 14 3 2017. [Online]. Available: https://news.aamc.org/medical-education/article/new-aamc-research-reaffirms-looming-physician-shor/. [Accessed 1 12 2017].

[5] T. Bollinger, "Benign and Malignant Tumors: What is the Difference?," 2017. [Online]. Available: https://thetruthaboutcancer.com/benign-malignant-tumors-difference/. [Accessed 4 12 2017].

[6] B. Waidyawansa, "Using the Wisconsin breast cancer diagnostic data set for predictive analysis," 3 12 2016. [Online]. Available: https://www.kaggle.com/buddhiniw/breast-cancer-prediction/notebook. [Accessed 5 12 2017].

[7] OpenCV, "Introduction to Support Vector Machines," 14 12 2017. [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html. [Accessed 14 12 2017].

[8] http://scikit-learn.org, "AdaBoost," 2017. [Online]. Available: http://scikit-learn.org/stable/modules/ensemble.html#adaboost. [Accessed 14 12 2017].

[9] scikit-learn.org, "Random Forests," 2017. [Online]. Available: http://scikit-learn.org/stable/modules/ensemble.html#forest. [Accessed 14 12 2017].