- **Naive Bayes -**
    1. Easy to predict class of test data even if dataset is large for multiclass prediction.
    2. If we assume independence amongst features then it performs better than logistic regression. But in wisc dataset case features are interdependent thus it performs best case equal to logistic regression.
    3. It performs best for categorical input values. But in our case there are numerical variables, in such case the distribution is <u>assumed to be normal distribution</u>.
    4. As the dataset doesn't have discrete counts or binary features thus it's best to assume that features follow a normal distribution, So we use GaussianNB. Which actually yields 3 folds better performance than Bernoulli or Multinomial NB.
    5. Ensembling, Boosting or Bagging is not advisable because Naive Bayes doesn't depend on variance, so reduction in variance doesn't effect.

- **Logistic Regression -**
    1. Able to produce nonlinear decision boundary.
    2. Unlike Naive Bayes it doesn't get effected by mild noise in dataset like multi-collinearity. Although we can use L2 regularization for High multi-collinearity for Gaussian prior due to less tolerance for large and small values for features.
    3. It doesn't handle large number of categorical features very well.
    4. It relies on transformation for non linear features.
    5. In the dataset the problem arises for logistic regression to large number of features but insufficient dataset. Thus model tends to fit the data almost perfectly. So we used a penalty for large parameter values. $\lambda\Sigma\theta_j^2$ is the usual function used for penalizing but scikitlearn gives a Inverse Regularization term C is basically 1/ λ. But no effect of Regularization is observed.

- **K Nearest Neighbors -**
    1. <u>KNN doesn't require any learning.</u> As the model for the only model for KNN is to store the entire dataset. Other efficient ways to store data is k-d trees.
    2. Most features have same metrics thus it's wise to use Euclidean distance measure. Else we would use Manhattan distance using distance between real vectors using the sum of their absolute distance.
    3. We can use a stochastic approach for very large datasets for which we will apply KNN.
    4. It is a lazy learning , Non parametric algorithm thus it doesn't make assumption about the distribution of instances.
    5. At k = 1 KNN obviously overfits the data and optimizes at 0.97 accuracy after 5+ neighbors. Notably accuracy drops with spikes and then after sharp changes gradually starts decreasing. On increasing the value of k more than 25.
    6. KNN can undergo the problem of Curse of Dimensionality due to large number of features in our dataset as in high dimensions points that maybe similar may have large distances as even if one out of 30 features varied the distance will increase abruptly. Thus it is mandatory to tune neighbors parameter for accurate results.

- **SVM -**
    1. As the dataset in unbalanced we try adjusting the reverse regularization term. And it gives optimal results at C = 6.
    2. SVM is not scale invariant so we can expect to achieve better result by scaling data on the input vector to (0, 1) or standardize it to mean 0 and variance 1.
    3. On scaled data applying a polynomial kernel with degree 3 gives maximum accuracy with 0.986013986014.
    4. There can exist two types of problems for SVM algorithms Primal or Dual , If we solve Primal problem we require computing mapping for each data point. But in dual formulation we compute the kernel function directly instead of the ordinary dot product. As we are using kernels thus the ratio of number of data points to the features matters and it's obvious to set it to False as in our dataset n_samples > n_features.
- **Decision Tree -**
    1. Decision Trees require least tuning, and does automatic feature selection. And it can also deal with noisy or incomplete data.
    2. But it has high classification error rate for more number of classes. And for large dataset calculation growth is exponential.
    3. In our dataset on analysing decision tree we see that nearly 75% datasets are classified as malignant or benign on basis of worst_perimiter.
    4. Without pruining dataset the tree overfits, thus we apply prepruining for more accurate results. And it yields better results at max_depth = 2.

    - **Ensembles of Decision Trees -**
      Ensembles are methods that combine multiple machine learning models.
        1. **Random Forests -**
            a. It removes the main drawback of overfitting of training data by Decision Trees.
            b. We average the results of various trees, whereas each tree might overfit to the separate training data but on averaging the amount of overfitting is reduced.
            c. Results greatly vary with the number of trees. For bootstrap sample of data we use max_features parameter's value to 'auto' which sets max_features to sqrt(30).
            d. High max_features means all trees will be quite similar and they will be able to fit the data easily. And low max_features will make the trees in random forest different and each tree might need to be very deep in order to fit the data.
            e. We use 'soft voting' strategy between outputs of all trees.
            f. On plotting feature_importance graph we see that all features are given non zero importance.
            g. We don't need to scale the data for using this algorithm.
            h. Dataset isn't sparse or very high dimensional so Random Forests is expected to give great results.
            i. The only drawback seems to be high memory usage and slow training and prediction.

2. **Gradient Boosting -**
    a. In gradient boosting all trees are built in a serialized manner and each succeeding tree tries to correct the mistakes of the previous tree.
    b. These require sensitive parameter tuning. As it is a regression method, learning rate influences how strongly each succeeding tree tries to correct the previous tree.
    c. In absence of lower learning rate or max_depth gradient boosting is likely to overfit thus we lower the learning rate to 0.001 (default 0.1) and try to optimize max_depth, at max_depth = 2 it gives best results.
    d. Feature importance in gradient boosted trees is like random forests but it ignores some of the features completely.

Both of the Ensembles of Decision trees work fine on the dataset, we can use any of these but Gradient Boosting is a better choice for large scale datasets.

Results on preprocessed features - (Training - 75%)

1. **SVM -**
    a. **Linear Kernel - 0.951048951049 @ C = 4, C doesn't effect much. (Unscaled Data)**
    b. **Polynomial Kernel - 0.986013986014 @ Degree = 3. (Scaled Data, Variance = 1, Mean = 0)**
2. **Decision Trees -**
    a. **Normal - 0.923076923077 @ max_depth = 5**
    b. **Random Forest - 0.993006993007 @ random_state = 2, max_features = sqrt(30), min_samples_split = 2, n_estimators = 11**
    c. **Gradient Boosting - 0.986013986014 @ learning_rate = 0.01 , max_depth = 2**
3. **Logistic Regression - 0.951048951049 @ C = 3**

4. **KNN - 0.951048951049 @ neighbors = 11, distance_metric = euclidean**

5. **Naive Bayes - 0.965034965035 @ Classifier_type = Gaussian Classifier**