Machine Learning Engineer Nanodegree

Capstone Project

Etienne Boshoff
July 16th, 2017

I. Definition

Project Overview

A Fine needle aspiration cytology (FNAC) is a diagnostic procedure in which a sample is obtained from a suspicious lump or mass of a patient by using a fine needle. A cytology test is then performed on the sample to determine if the lump or mass is benign or malignant. The FNAC is a commonly used procedure as it does not require the patient to be hospitalized. Anesthesia is not required either. The procedure is minimally invasive, economical and allows for rapid and accurate diagnosis. The final diagnosis is generally available in about a week.

However, initial results for some cases are non-definite. To reach a diagnosis, it may be necessary for the FNAC procedure is to be evaluated along with a histology and/or a radiological test. It is also important that the FNA biopsy be performed by an experienced practitioner to guaranty a valid sample. But even if performed correctly, FNAC is believed to miss between 5 and 10 percent of cancers.

One possible solution to improve the FNAC process in producing a diagnosis, is to use a microscope to identify specific characteristics from the FNA sample and to feed this information into a Machine Learning classification model that would give an instant cancer diagnosis prediction which can later be corroborated with the cytology diagnosis.

This would allow for cancer treatments to be planned and prescribed earlier, reduce the time the patient needs to wait anxiously for the diagnosis and can help to validate the sample when in doubt.

This idea of predicting a diagnosis by applying a machine learning algorithm on certain microscopically visible cell characteristics of a FNA sample came from Dr. William H. Wolberg (General Surgery Dept. at the University of Wisconsin). He identified nine visually assessed characteristics of FNA samples he considered relevant to reach the diagnosis. These characteristics were identified for several cases along with the definitive diagnosis. One of the resulting data sets is the well-known Breast Cancer Wisconsin (Diagnostic) Data Set. Many scholars have used this data set to create machine learning models to predict cancer diagnosis.

Problem Statement

In this project, a supervised machine learning binary classification model will be developed. It will be trained and tested on the Breast Cancer Wisconsin (Diagnostic) Data Set. The input features were

obtained by visually assessing the cellular nuclei from stained samples on microscope slides. The output of the model will be the prediction of the cancer diagnosis (Benign or Malignant).

Metrics

The model to be created is expected to predict if a patient has cancer or not. It is therefore very important for the model to have a high predictive performance. The higher the estimation of the predictive performance, the more useful the model will be in assisting healthcare providers in reaching diagnoses.

The predictive performance is estimated on a test set which represents unseen future data. The estimation is done by scoring the differences between the actual test set labels and the labels predicted by the model. The 3 basic scores typically used for estimating predictive performance are: accuracy, precision and recall.

Intuitively, accuracy relates to the proportion of correct results that the model achieved. Because of how the accuracy is calculated, a model's accuracy can increase even if false negatives or false positives are increased. This characteristic makes the accuracy not an appropriate score on its own to estimate predictive performance as special attention must be placed to keep false negatives and false positives low. Precision and recall are better options to focus on in this project. Precision answers the following question: out of all the samples the model labeled as positive, what fraction were correct? On the other hand, recall answers the question: out of all the positive samples there are, what fraction did the classifier pick up?

The formulas for the 3 mentioned scores are:

```
accuracy = (TP + TN)/(TP + TN + FP + FN)
precision = TP/(TP + FP)
recall = TP/(TP + FN)
```

Ideally all three scores are to be as high as possible. However, training a model that has a score of 1 for both precision and recall is not always possible. Normally, a model is trained to either increase in precision or in recall. As false negatives are considered more dangerous than false positives (as false negatives can cause a patient with cancer to be sent home under the believe to not have cancer), the main scoring metric selected to evaluate model predictive performance for this project will be recall.

II. Analysis

Data Exploration

The following table displays for each feature in the data set, basic statistics (columns *Count, Mean, Std., Min., 25%, 50%, 75% and Max.*) and the type of data once imported into a *pandas* data frame (column *DType*). The table also contains a sample (column Sample):

Feature	Sample	Count	Mean	Std.	Min.	25%	50%	75%	Max.	Dtype
id	84300903	569	3E+07	1E+08	8670	869218	906024	8813129	9E+08	int64
diagnosis	M	569	N/A	N/A	N/A	N/A	N/A	N/A	N/A	O
radius_mean	19.69	569	14.127	3.524	6.981	11.7	13.37	15.78	28.11	float64
texture_mean	21.25	569	19.29	4.301	9.71	16.17	18.84	21.8	39.28	float64
perimeter_mean	130	569	91.969	24.3	43.79	75.17	86.24	104.1	188.5	float64
area_mean	1203	569	654.89	351.9	143.5	420.3	551.1	782.7	2501	float64
smoothness_mean	0.1096	569	0.096	0.014	0.053	0.086	0.096	0.105	0.163	float64
compactness_mean	0.1599	569	0.104	0.053	0.019	0.065	0.093	0.13	0.345	float64
concavity_mean	0.1974	569	0.089	0.08	0	0.03	0.062	0.131	0.427	float64
concave points_mean	0.1279	569	0.049	0.039	0	0.02	0.034	0.074	0.201	float64
symmetry_mean	0.2069	569	0.181	0.027	0.106	0.162	0.179	0.196	0.304	float64
fractal_dimension_mean	0.05999	569	0.063	0.007	0.05	0.058	0.062	0.066	0.097	float64
radius_se	0.7456	569	0.405	0.277	0.112	0.232	0.324	0.479	2.873	float64
texture_se	0.7869	569	1.217	0.552	0.36	0.834	1.108	1.474	4.885	float64
perimeter_se	4.585	569	2.866	2.022	0.757	1.606	2.287	3.357	21.98	float64
area_se	94.03	569	40.337	45.49	6.802	17.85	24.53	45.19	542.2	float64
smoothness_se	0.00615	569	0.007	0.003	0.002	0.005	0.006	0.008	0.031	float64
compactness_se	0.04006	569	0.025	0.018	0.002	0.013	0.02	0.032	0.135	float64
concavity_se	0.03832	569	0.032	0.03	0	0.015	0.026	0.042	0.396	float64
concave points_se	0.02058	569	0.012	0.006	0	0.008	0.011	0.015	0.053	float64
symmetry_se	0.0225	569	0.021	0.008	0.008	0.015	0.019	0.023	0.079	float64
fractal_dimension_se	0.004571	569	0.004	0.003	0.001	0.002	0.003	0.005	0.03	float64
radius_worst	23.57	569	16.269	4.833	7.93	13.01	14.97	18.79	36.04	float64
texture_worst	25.53	569	25.677	6.146	12.02	21.08	25.41	29.72	49.54	float64
perimeter_worst	152.5	569	107.26	33.6	50.41	84.11	97.66	125.4	251.2	float64
area_worst	1709	569	880.58	569.4	185.2	515.3	686.5	1084	4254	float64
smoothness_worst	0.1444	569	0.132	0.023	0.071	0.117	0.131	0.146	0.223	float64
compactness_worst	0.4245	569	0.254	0.157	0.027	0.147	0.212	0.339	1.058	float64
concavity_worst	0.4504	569	0.272	0.209	0	0.115	0.227	0.383	1.252	float64
concave points_worst	0.243	569	0.115	0.066	0	0.065	0.1	0.161	0.291	float64
symmetry_worst	0.3613	569	0.29	0.062	0.157	0.25	0.282	0.318	0.664	float64
fractal_dimension_worst	0.08758	569	0.084	0.018	0.055	0.071	0.08	0.092	0.207	float64

Dataset and inputs

The dataset contains a total of 32 features and 569 instances. The real-valued features [rows #3 - #32 from the table above] describe characteristics of cell nuclei and were computed from digitized images of breast mass samples obtained by FNA procedures.

The features of the dataset are described as follows:

- **ID** number for each sample (row #1 from the table). This feature will be removed as it does not add value to the ML model.
- **Diagnosis** (M = malignant, B = benign) (row #2 from the table). This is the output variable (label) of the dataset.
- The remaining features corresponds to the *mean* [row #3 – row #12 from the table], *standard error* [row #13 – row #22] and *"worst" or largest (mean of the three largest values)* [row #23 – row #32] for the following ten-real valued characteristics for each cell nucleus:

    - **radius** (mean of distances from center to points on the perimeter)
    - **texture** (standard deviation of gray-scale values)
    - **perimeter**
    - **area**
    - **smoothness** (local variation in radius lengths)
    - **compactness** (perimeter^2 / area - 1.0)
    - **concavity** (severity of concave portions of the contour)
    - **concave points** (number of concave portions of the contour)
    - **symmetry**
    - **fractal dimension** ("coastline approximation" - 1)

Missing data

From the count column in the table above, we can see there are no missing values for any of the features in the dataset: all features have exactly 569 values.

Categorical data

All input features from the dataset are of type *float64* except for the ***Id*** feature, which will be removed as it does not add value to the model. The feature containing the class labels (***diagnosis***) is nominal (values B and M). Encoding will be used to convert its values into numerical values (0 and 1).

Feature scaling

From the table above, it can be noted that not all features are on the same scale. For example, feature ***area_worst*** (measured from 185.2 to 4254) is on a much larger scale than feature ***smoothness_se*** (measured from 0.002 to 0.031).

Outlier detection

From all the values of a feature, the outliers are values considered to be much greater or smaller than the rest. The presence of outliers can mislead machine learning algorithms during training causing the algorithms to have low predictive performance.

For this project, Tukey's Method is used for identifying outliers. It does not assume the features to be normally distributed and it ignores the mean and standard deviation which makes it resistant to being influenced by the extreme values in the range. In this method, an *outlier step* is calculated as 1.5 times the interquartile range (IQR). A data point with a feature that is beyond an outlier step outside of the IQR for that feature is considered abnormal.

Applying Tukey's Method on the already standardized dataset, all features except ***concave points_worst*** are identified to have a substantial amount of outlier values. Even multiplying the interquartile range (IQR) by 3 instead of 1.5 to identify extreme outliers, the method identifies 23 features with at least one outlier value.

There are 2 instances with an extreme outlier value for 6 features, 4 instances with an extreme outlier value for 5 features, 9 instances with an extreme outlier value for 4 features, 16 instances with an extreme outlier value for 3 features, 32 instances with an extreme outlier value for 2 features and there are 56 instances with an extreme outlier value for one feature.

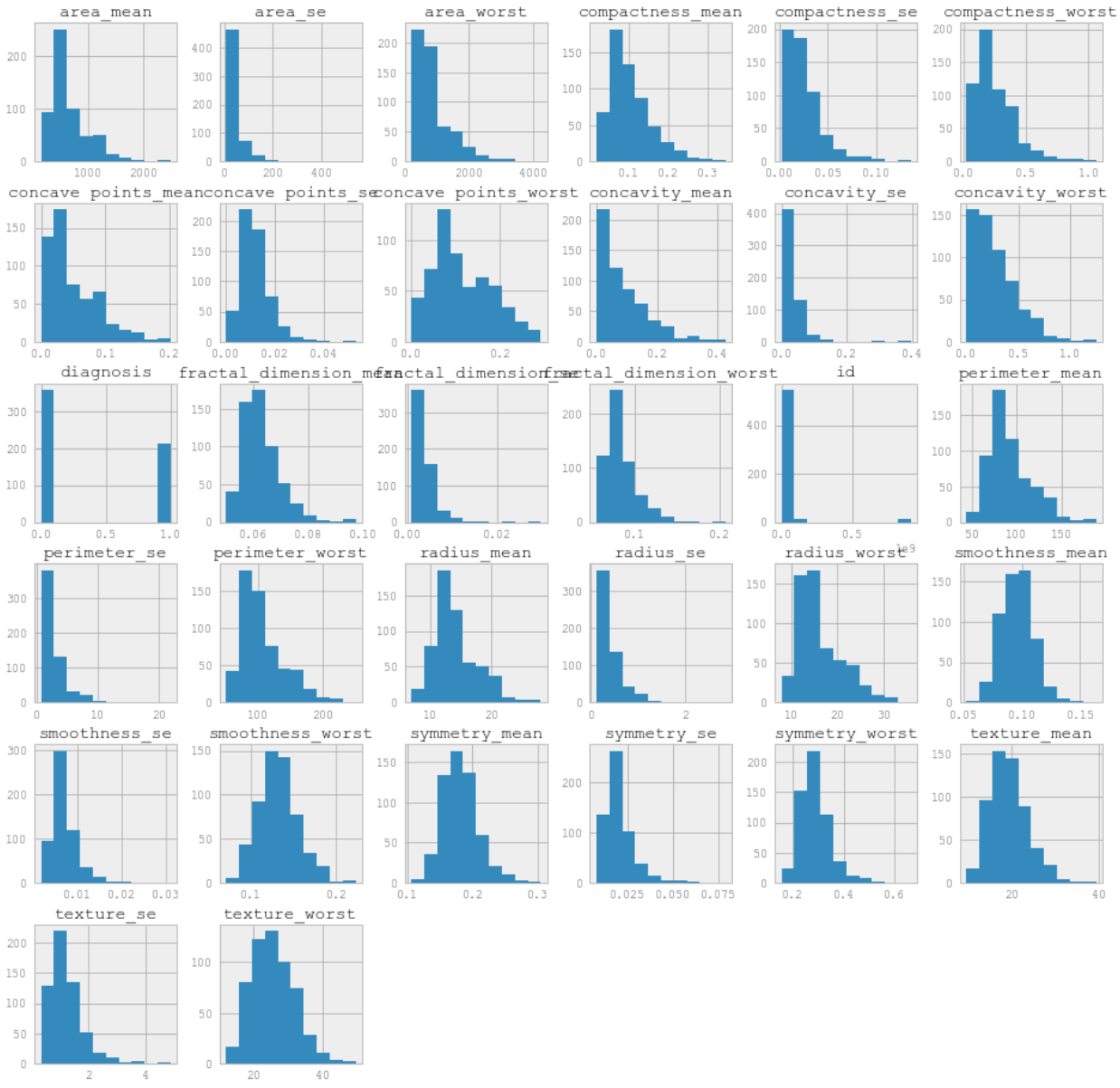
Class balance

There are 357 instances for class B and 212 instances for class M. The ratio is 1 to approximately 1.683. This indicates a slight class imbalance in the dataset.

Exploratory Visualization

Skewness of features

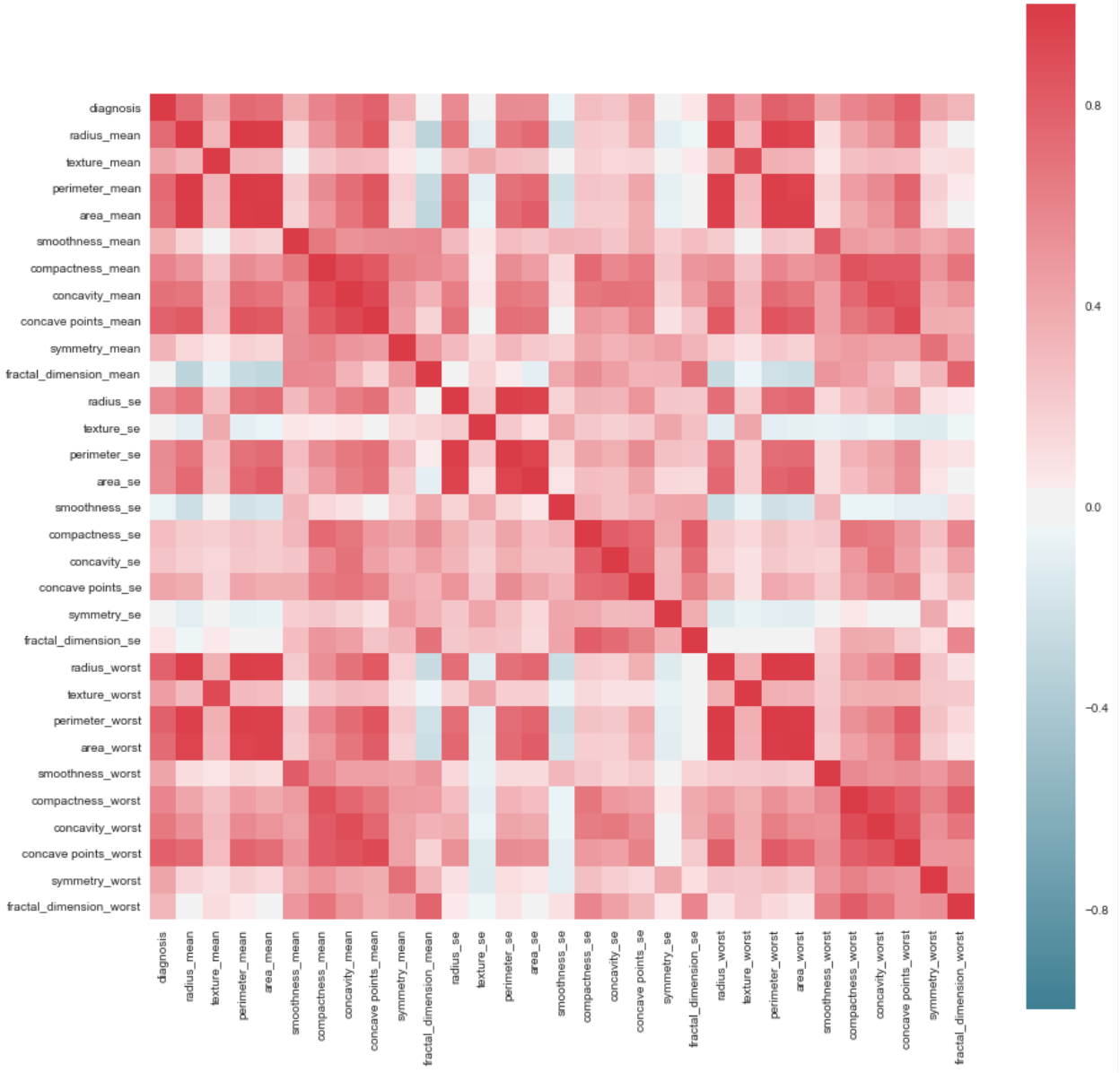
Skewness is a measure of asymmetry for a distribution. Highly-skewed feature distributions can negatively affect the performance of learning algorithms. The following plot shows the distribution graph for each input feature:



Each graph can be visually evaluated to determine the skewness of the corresponding feature. As can be noted from the graphs, many distributions are significantly skewed. The most skewed features are ***concavity_se*** and ***area_se***. The less skewed features are ***smoothness_worst***, ***smoothness_mean***, ***points_worst*** and ***texture_worst***.

Feature Correlation

The following plot shows the heatmap for the dataset. The heatmap is a matrix that displays how correlated each feature is with the other features:



Algorithms and Techniques

Techniques used for pre-processing the data:

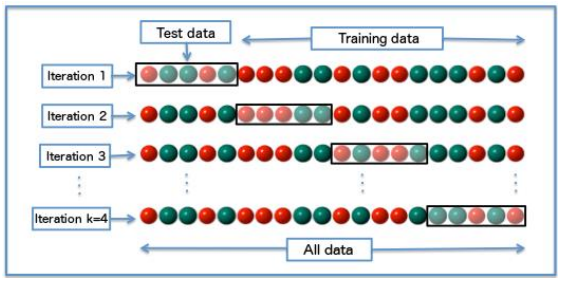
- LabelEncoder: This technique is used to convert the categorical label values from “M” and “B” into the numerical values 0 and 1. Most algorithms expect numerical labels.
- StandardScaler: This technique is applied to solve the problem of features being on different scales and the problem of some features being skewed. Both characteristics can negatively impact the predictive performance of learning algorithms. The technique transforms the values of each feature to have a mean of 0 and a standard deviation of 1 to give each feature the form of a normal distribution and for all features to be on a similar scale.

Techniques used for splitting the data:

- Train_test_split: This technique is used to split the dataset into a training set and a test set.
- StratifiedKFold: This technique is used to split the training set into folds when cross validation is used. In order to select the best classifier and to fine tune it, different models are trained and tested on different groupings of the folds. This technique tries to maintain the original class balance for each fold to avoid generating any folds in which the imbalance ratio is worse than in the original dataset.

Technique used for managing scores:

- Cross_val_score: this technique uses the StratisfiedKFold technique to obtain different folds. It iterates through different groupings of training and test folds. For each iteration, it trains a model on the corresponding training folds put together, tests the model on the corresponding testing fold and stores the testing score in an array. The result is the array containing the testing score for each iteration. This technique is used for selecting the most appropriate classifier algorithm and also to fine tune it. Classifier algorithms and fine-tuning hyperparameters are evaluated based on the average scores obtained. This is to avoid scores being over sensitive to any particular training and test sets randomly selected.



By Fabian Flöck - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=51562781

The following supervised learning algorithms for classification are evaluated on the training set using cross validation. The algorithm that generates models with the highest average scores is selected as the final model to be fine-tuned.

- Decision Tree: it makes less assumptions on the training data than other algorithms. If during data analysis a characteristic is missed that should be dealt with in pre-processing, the algorithm’s predictive performance is expected to be less affected. However, Decision Tree models tend to overfit

and to be biased towards the dominating class when the data set is unbalanced. Bad predictive performance will indicate either the need to balance the dataset or that the problem is not trivial to solve. Additionally, this algorithm is expected to be a benchmark for the Adaboost algorithm also being considered.

- Adaboost: it is expected to outperform the Decision Tree model. The Adaboost algorithm creates models that are less sensitive to the class unbalance. Bad predictive performance could be an indication of noise in the data, the existence of outliers not dealt with or that dimensionality reduction is required.
- SVM: its predictive performance will be much less effected if the number of features is high in relation to the number of samples. SVMs are not part of the Decision tree family and could be a good alternative in case the decision tree family algorithms are not appropriate for this dataset.
- KNN: the predictive performance of the KNN algorithm will be less impacted by any noise in the data set. It is expected to outperform if the decision boundary is very irregular.
- GaussianNB: it is being used as the benchmark model for the final supervised classification model to be generated in this project. It assumes features not to be correlated and to be normally distributed. It is not expected to have a high predicting score for the dataset as features are highly correlated and several features are not normally distributed.

Technique used for finding the most appropriate hyperparameters:

- GridSearchCV: this technique identifies the most appropriate combination of hyperparameters which would generate the final fine-tuned model. It uses cross validation on the training set to obtain an average score for each combination of hyperparameters.

Benchmark

The benchmark model will be the point of reference to determine if the fine-tuned model adds any significant value. For this project, the GaussianNB supervised classifier is being used to generate the benchmark model.

A GaussianNB model is an improvement over the naïve model of predicting the dominating class for all samples. It is expected to be negatively affected by the features being skewed and highly correlated. Any fine-tuned model must clearly outperform this benchmark model.

The scores obtained by the benchmark model on the test set are the following:

```
Accuracy on Test set: 0.912
Precision on Test set: 0.864
Recall on Test set: 0.905
Confusion Matrix - Predictions on Test set:

Predicted    0   1  All
True
0           99   9  108
1            6  57   63
All        105  66  171
```

The confusion matrix shows that there are 6 false negatives and 9 false positives. The fine-tuned model must reduce both false negatives and false positives (especially false negatives as these errors can lead to delay in treatments).

III. Methodology

Data Preprocessing

Dealing with missing data

The data set does not contain missing data. No action is required.

Handling categorical data in the input features

The data set does not contain categorical data in the input features. No action is required.

Encoding class labels

The categorical label values “M” and “B” are encoded into the numerical values 0 and 1.

Removing features that don’t add meaningful information

Feature ***Id*** is removed as its only purpose is to identify the samples. It does not add any valid information during training or testing.

Separate input features and labels

The diagnosis feature is removed from the features and stored into a separate data frame. The remaining features define the input features.

Standardization

Input features are standardized to bring values of all features onto the same scale and for features to be normally distributed.

Partitioning the dataset into training and test sets

The dataset is split to have 30% of its samples in the test set and 70% in the training set.

Class imbalance

The ratio of class Malignant to class Benign is 1:1.6. This is a very slight imbalance and is not expected to have a significant impact on all algorithms considered. Instead of using a technique like SMOTE to oversample the underrepresented class, StratifiedKFold is used during cross validation to avoid obtaining folds with a higher imbalance ratio.

Feature correlation

Because of the curse of dimensionality, if the number of features is high in relation to the number of samples, certain models could suffer from overfitting. If overfitting was occurring because of high dimensionality, either more samples would need to be added to the dataset or a technique for reducing dimensionality would need to be applied. In case any models suffered from overfitting, removing correlated features would be a good option for dimensionality reduction. As overfitting is not a problem

for any of the models being evaluated, a feature reduction technique (like LDA or feature selection) is not applied.

Dealing with outliers

As seen in the Exploration section, Tukey's method was used to identify outliers. A very high number of outliers were identified, even when considering extreme outliers.

Removing this number of outliers would not be appropriate as the dataset would be significantly reduced. An option would be to remove samples containing extreme outliers for more than one feature.

However, it is preferred not to remove any samples as there is no indication to believe that any values are recorded in error. No specific action is taken to deal with outliers.

Implementation

As mentioned, the goal is to generate a model with the best possible predictive performance on future (unseen) data, having trained and tested the model on the cancer dataset.

In order to identify the supervised classification algorithm that would generate the model with the best predictive performance, multiple algorithms are evaluated and compared. A model with default hyperparameters is generated with each candidate algorithm and the predictive scores of each model are then compared to select the algorithm.

The algorithms considered as candidates are the following: SVM, KNN, Decision Tree, AdaBoost and GaussianNB. All these algorithms are popular algorithms and appropriate for the given goal and dataset.

The initial approach was to split the dataset into a training set and a test set, train a fine-tuned model with each candidate algorithm on the training set and then compare the accuracy scores for each fine-tuned model on the test set to choose the final solution. The challenge of this approach was that a different algorithm and hyperparameters were being identified as most appropriate almost every time a new training and testing set was generated. The final approach is to first identify the most appropriate algorithm (considering default hyperparameters) using cross validation on the dataset. Once the most appropriate algorithm is identified, the model is then fine-tuned.

The first step in selecting the algorithm is to define a model using default hyperparameters for each of the considered algorithms:

```
clf = SVC(random_state=0)
knn = KNeighborsClassifier()
bdt = AdaBoostClassifier(random_state=0)
gnb = GaussianNB()
```

Cross validation is used to avoid the scores from being too sensitive to any specific splits of the dataset. This technique is used to split the entire preprocessed dataset into 8 folds. This number of folds is considered appropriate taking into consideration the number of samples in the dataset. As the dataset is slightly unbalanced, the folds are made by maintaining the percentage of samples for each class. This is to avoid any of the folds having a higher unbalance that could affect scores. Data is shuffled before each split is done.

For the folds to be generated as described above, the following *Stratified K-folds Converter* is defined:

```
skf = StratifiedKFold(n_splits=8, shuffle=True, random_state=0)
```

For each model, the 8 folds are generated on the training set. The model is evaluated on each fold having been trained on the remaining folds put together. The score for each fold is stored in an array. Function *cross_val_scorer* is used to train and obtain the score arrays:

```
scores_clf = cross_val_score(clf, X_train, y_train, cv=skf, scoring='recall')
scores_knn = cross_val_score(knn, X_train, y_train, cv=skf, scoring='recall')
scores_bdt = cross_val_score(bdt, X_train, y_train, cv=skf, scoring='recall')
scores_gnb = cross_val_score(gnb, X_train, y_train, cv=skf, scoring='recall')
scores_dtc = cross_val_score(dtc, X_train, y_train, cv=skf, scoring='recall')
```

The *mean* and *std*. is calculated for the results (arrays) returned from the *cross_val_score* function for each model:

```
print("SVM Recall score: %0.3f (+/- %0.3f)" % (scores_clf.mean(), scores_clf.std() * 2))
print("KNN Recall score: %0.3f (+/- %0.3f)" % (scores_knn.mean(), scores_knn.std() * 2))
print("BDT Recall score: %0.3f (+/- %0.3f)" % (scores_bdt.mean(), scores_bdt.std() * 2))
print("GNB Recall score: %0.3f (+/- %0.3f)" % (scores_gnb.mean(), scores_gnb.std() * 2))
```

The Recall score metric is being considered to compare the models. Typical metrics considered for scoring is Precision and Recall. High precision relates to a low false positive rate and a high recall relates to a low false negative rate. Both false positives and false negatives are to be minimized. However, false negatives are especially non-desirable as they lead to treatment being delayed or not prescribed at all (as patient could be sent home under the believe to be free from cancer). So, when models are being compared, focus is placed on the Recall (low false positive rate) metric.

The mean and std. values obtained for each model are the following:

```
SVM Recall score: 0.967 (+/- 0.059)
KNN Recall score: 0.929 (+/- 0.078)
BDT Recall score: 0.925 (+/- 0.098)
GNB Recall score: 0.896 (+/- 0.091)
```

Random states are used for results to be the same each time the code is run. In production, the ***random_state*** parameter would be removed. In any case, not using the random state still identifies the SVM model for every run.

As the above results show, SVM is the model that consistently produces the highest mean of scores. Its std. is very small and so we know all scores are similar to each other. Therefore, SVM is the algorithm chosen to generate the model to be fine-tuned.

Refinement

As mentioned in the previous section, the SVM supervised classification algorithm has been identified as the algorithm that generates the model with the highest prediction performance. The model was obtained by considering the default hyperparameters of the algorithm.

The next step is to obtain the combination of hyperparameters that would allow the SVM algorithm to produce a fine-tuned model in order to improve the predictive performance obtained earlier.

Different values for the hyperparameters are defined as a list of dictionaries:

```
# Set the hyperparameters to be used for the GridSearch
tuned_parameters = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
                     'C': [1, 10, 20, 30, 100, 200, 300]},
                    {'kernel': ['linear'], 'C': [1, 10, 20, 30, 100, 200, 300]}]
```

Just as when the base models were obtained and compared, special care must be taken to prevent the fine-tuning to be over sensitive to a specific training set. As before, cross validation is used. For each combination of hyperparameter values, the training set is split into 8 folds and for each of the folds obtained, the recall score is calculated after having trained the model on the remaining of the folds put together. The GridSearchCV function implements this process. It receives the *Stratified K-folds Converter* (object skf) which defines that the 8 folds must be obtained for each stratification and that the folds must be made by maintaining the original percentage of samples for each class:

```
clf = GridSearchCV(SVC(C=1), tuned_parameters, cv=skf, scoring='recall_macro')
clf.fit(X_train, y_train)
```

For each hyperparameter value combination, the recall scores obtained from each fold are stored into an array. The mean and std. are obtained and compared. Some results obtained are:

```
0.933 (+/-0.098) for {'kernel': 'rbf', 'C': 1, 'gamma': 0.001}
0.638 (+/-0.063) for {'kernel': 'rbf', 'C': 1, 'gamma': 0.0001}
0.956 (+/-0.075) for {'kernel': 'rbf', 'C': 10, 'gamma': 0.001}
0.930 (+/-0.111) for {'kernel': 'rbf', 'C': 10, 'gamma': 0.0001}
0.963 (+/-0.056) for {'kernel': 'rbf', 'C': 20, 'gamma': 0.001}
0.938 (+/-0.105) for {'kernel': 'rbf', 'C': 20, 'gamma': 0.0001}
0.968 (+/-0.052) for {'kernel': 'rbf', 'C': 30, 'gamma': 0.001}
0.936 (+/-0.090) for {'kernel': 'rbf', 'C': 30, 'gamma': 0.0001}
0.971 (+/-0.042) for {'kernel': 'rbf', 'C': 100, 'gamma': 0.001}
0.956 (+/-0.075) for {'kernel': 'rbf', 'C': 100, 'gamma': 0.0001}
0.964 (+/-0.048) for {'kernel': 'rbf', 'C': 200, 'gamma': 0.001}
0.963 (+/-0.056) for {'kernel': 'rbf', 'C': 200, 'gamma': 0.0001}
0.961 (+/-0.055) for {'kernel': 'rbf', 'C': 300, 'gamma': 0.001}
0.968 (+/-0.052) for {'kernel': 'rbf', 'C': 300, 'gamma': 0.0001}
0.965 (+/-0.055) for {'kernel': 'linear', 'C': 1}
0.969 (+/-0.052) for {'kernel': 'linear', 'C': 10}
0.963 (+/-0.056) for {'kernel': 'linear', 'C': 20}
0.959 (+/-0.054) for {'kernel': 'linear', 'C': 30}
```

The hyperparameter value combination that generated the fine-tuned model with the highest recall score is:

```
{'kernel': 'rbf', 'C': 100, 'gamma': 0.001}
```

The fine-tuned model is now used to predict labels for the testing set. Accuracy, precision and recall scores are obtained. The confusion matrix for the model's prediction on the test set is also obtained:

```
Accuracy on Test set: 0.982
Precision on Test set: 0.969
Recall on Test set: 0.984
Confusion Matrix - Predictions on Test set:

Predicted    0   1  All
True
0          106   2  108
1            1  62   63
All        107  64  171
```

From the test set, there were only 1 false negative and 2 false positives. As expected, we have a higher recall than precision.

IV. Results

Model Evaluation and Validation

Before the algorithm is selected and fine-tuned, the data set is randomly split into a training set and a test set. The testing set represents unseen future data and allows to evaluate how good the predictive performance of the model will be on future data.

To identify the SVM supervised classification algorithm as the most appropriate algorithm for this problem, the technique of cross validation was used on the training set for all the candidate algorithms. This means that the algorithm was selected based on an average score across several random splits of the training set. Therefore, the algorithm selection does not directly depend on a single specific training set. The algorithm was selected because it generates the model with the highest average score across each fold having been trained on the remaining folds put together.

Cross validation was also used to fine-tune the algorithm on the training set. As before, the hyperparameter values are not selected based on one single set. A model is created for each combination of hyperparameter values and is trained and scored based on different folds. This helps to guarantee obtaining the best hyperparameters. The fine-tuning is therefore not sensitive to variations in the training set.

After the model is trained, it is validated on the test set to evaluate how it will behave on a set it has never seen before. As the process has never been in contact with the test set, its behavior on the test set is a valid prediction on how the model will behave on future (unseen) data.

The accuracy, precision and recall scores on the test set are very high. The recall score obtained is 0.984. We are especially interested in the recall score as it relates to low false negatives. Having low false negative rates is very important for the expected solution as false negatives can lead to treatments being delayed. From the confusion matrix, it can be noted there are only 1 false negative and 2 false positives.

These scores indicate the model generalizes very well on unseen data and it can be confirmed the model does not overfit.

Justification

The benchmark model is a GaussianNB model with default hyperparameters. It was trained on the testing set and its scores on the test dataset are the following:

```
Accuracy on Test set: 0.912
Precision on Test set: 0.864
Recall on Test set: 0.905
Confusion Matrix - Predictions on Test set:

Predicted    0   1  All
True
0           99   9  108
1            6  57   63
All        105  66  171
```

These results show that the benchmark model had already reached high scores even though there are room for improvement. Out of the 63 patients with a malignant diagnosis in the test set, the model predicted no cancer for 6 patients. This rate needs to be improved as this model could have potentially delayed treatment for those 6 patients (out of the 63 total patients with cancer).

The final model being proposed is a SVM model with hyperparameters:

```
{'kernel': 'rbf', 'C': 100, 'gamma': 0.001}
```

The scores for this mode are the following:

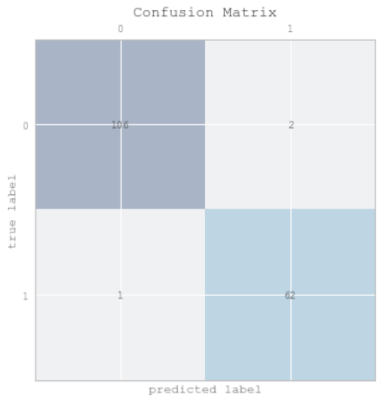
```
Accuracy on Test set: 0.982
Precision on Test set: 0.969
Recall on Test set: 0.984
Confusion Matrix - Predictions on Test set:

Predicted    0   1  All
True
0          106   2  108
1            1  62   63
All        107  64  171
```

These results show that the final model is an improvement over the benchmark model. All its scores are higher than those of the benchmark model. We were especially interested in obtaining a model with a higher recall score. The final model's recall score is 0.984 whereas the benchmark model's recall score is 0.905. From the confusion matrix, the final model only produced one false negative out of the 63 patients with a positive cancer diagnosis.

V. Conclusion

Free-Form Visualization

The following is a plot of the confusion matrix obtained by comparing the predicted labels to the real labels on the test set:



It shows there are 106 True Negatives, 62 True Positives, 2 False Positives and 1 False Negative.

Reflection

The objective of this project is to generate a binary supervised classification model that predicts whether a lump from a patient is benign or malignant based on certain microscopically viewable characteristics of a stained sample obtained from the lump through a FNA procedure.

To create the model, a supervised classifying algorithm is chosen from a group of four pre-selected algorithms. The selected algorithm is fine-tuned to generate the final model which is compared to a benchmark model.

The process starts with the *exploration* of the dataset. The following was determined during exploration:

- The dataset has no missing values.
- All input features are numerical.
- There is a slight class unbalance towards the Benign class with a ratio of 1:1.6.
- Not all features are on the same scale.
- Many features are significantly skewed.
- Many features are highly correlated.
- Data is quite noisy as many outliers were detected.

During data *pre-processing* phase, input features were standardized to bring all input feature values onto a similar scale and to give features a normal distribution. Class labels “B” and “M” were encoded to 0 and 1. Due to the high scores obtained for the final model, it was not believed necessary to deal with the class

unbalance and correlated features. It was also decided not to deal with the outliers because of the number of outliers detected and the origin of the data set. However, it was decided to maintain the class balance when folds are created during cross validation to avoid having folds with a higher unbalance.

For implementing the model, the data set was split into a training set and a test set. Cross validation was used to on the training set to generate several models for each considered algorithm. Each algorithm was evaluated by the average recall scores of the models created. The algorithm that consistently produces the highest average recall scores is SVM. Cross validation was also used for fine-tuning the SVM algorithm. Each combination of hyperparameters was evaluated by the average recall scores of the models created on different sets defined with cross validation.

Finally, the predictive performance of the fine-tuned model is compared to the benchmark model on the test set.

Some interesting aspects of the project:

- The number of extreme outliers identified was much higher than expected. There are 56 instances with an extreme outlier value for exactly one feature.
- All algorithms considered produced models with very similar scores. Additionally, default model scores were much higher than expected (over 89%). SVM produced the highest scores but was very closely followed by KNN, Decision Tree, Adaboost, GaussianNB and KNN.

Application of the model:

The final model produced has very high scores for accuracy, precision and recall:

```
Accuracy on Test set: 0.982
Precision on Test set: 0.969
Recall on Test set: 0.984
Confusion Matrix - Predictions on Test set:

Predicted    0   1  All
True
0          106   2  108
1            1  62   63
All        107  64  171
```

These high scores (specially the recall score) makes to model appropriate to be proposed as a valid tool to assist healthcare providers in detecting cancer. A software product could be developed to use the final model along with a system capable of automatically analyzing a FNA sample for capturing the value for each of the features the model is trained on. The results of this product should be compared to the final diagnosis reached with the current standard process for analyzing samples. Once it has been determined in which cases the model produces false positive or negative results, the product would be ready to be evaluated by the clinical community and entities like the FDA.

Improvement

At this point it is difficult to identify which improvements could be made to reduce the false positives and negative results to zero. More training data could be necessary or maybe more attention needs to be placed on detecting outliers. Additional machine learning algorithms could be considered.