Capstone Project - Proteomics of Breast Cancer

Machine Learning Engineer Nanodegree

Dom Sykes

October 25th, 2016

I. Definition

Project Overview

The problem I have chosen to focus this project on is in the area of Healthcare. Specifically it looks into proteomics, which is the study of the proteins expressed by a cell, and hopes to identify characteristics of the disease of Breast Cancer with the objective to provide diagnostic help.

I have chosen to do a project in this domain as I wish to engage in medical science but hadn't previously identified how I wished to get involved. While in the lab during my undergraduate degree it became apparent that while biology and medicine were becoming more apt at generating data, those they could process, manipulate and gather insights on it effectively were few. Through pursuing this project I hope to pick up a skill set that will allow me to effectively apply machine learning techniques to harness the data sets available in medical science.

Breast cancer is one of the most common forms of cancer, and affects one in eight women during their lifetime, killing thousands every year. One of the most common descriptions of cancer progression is the AJCC stages, which go from Stage I to Stage IV dependent on characteristics of the cancer such as tumor size, presence in the lymph nodes and if it has metastasised.

The data sets I shall be using are from Kaggle, and include two tables of data. The first is protein expression data for thousand proteins of each patient, with details of the gene from which these proteins originate. The second is a table of clinical data of the same patients, such as age, gender and the characteristics listed above.

Here is a link to the dataset:

https://www.kaggle.com/piotrgrabo/breastcancerproteomes

Cancer is essentially cells malfunctioning and mutating the pathways that are essential to life. Growth for example is key as cells are always dying and being replenished, however when a cell becomes cancerous this growth becomes uncontrolled and dangerous. The ways cancers mutate these pathways is through changing the proteins they express. Proteins are a crucial part of every cell, providing structure, messaging and the “doing” parts of the cell cycle, catalysing and controlling the chemical reactions crucial to life. Reducing proteins associated with suppressing growth, or increasing expression of proteins associated with promoting growth for example is one the telltale signs that cancer has taken hold. Identification of changes in expression of these proteins and other characteristics of cancer could lead to new, quicker, less time consuming diagnostics. Knowing how the cancer is affecting the cells could also lead to new, more targeted treatments.

Problem Statement

My problem statement is “Can protein expression levels help determine the stage of Cancer, including whether it has undergone metastasis, whether it's in the lymph nodes and tumor characteristics.”

To solve this I shall begin by matching the data from the two tables, changing name conventions and removing patients that don't exist in both. Following this I shall use PCA to reduce the number of features as this outweighs number of samples by many fold in the original data.

I shall run feature selection and machine learning algorithms choosing the best numbers of features for each algorithm and tuning the best of these for success rate in identifying the right AJCC stage for each patient.

Success for this question would be a model that provides a high reliability of predicting the stage of the cancer, using only the protein expression data.

Metrics

I shall be looking at a range of metrics including accuracy, recall and precision, to gauge the performance of my models. The main two I shall be using F1_score and cohen kappa score as it is a classification problem as well as being multi-class, and for success the model needs to be both accurate and precise.

F1_score is chosen as it combines both recall and precision to provide a more rounded measure than either on their own, or accuracy. This is particularly important when classes are unbalanced as they are here. F1 score is defined as 2(precision * recall)/(precision + recall), and has been found to be a more balanced measure for data sets of an abnormal distribution. Figure 1 shows the equation clearer.

Figure 1. F1 Score Equation

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

Precision is defined as the percentage of times a prediction class is correct, and it's formula is true positives / (true positives + false positives). Figure 2 shows clearer the equation, where TP is count of true positives and FP is count of false positives.

Figure 2. Precision Equation

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall is defined as the percentage of total instances in a dataset that a particular class is correctly identified. It's formula is true positives / (true positives + false negatives). Figure 3 shows clearer the equation, where TP is count of true positives and FN is count of false negatives.

Figure 3. Recall Equation

$$\text{Recall} = \frac{TP}{TP + FN}$$

As these are both percentages their values range from 0 to 1. Combining these two measures ensures F1 Score is a fairer reflection of the predictive power of the model.

Cohen kappa score is a measure to compare labelings by two different authors, with the two different authors in this case being the AJCC Stage and the labeling of my models. While these two authors are certainly not completely independent, as my models are learning from the AJCC Stage, this measure is being used due to it being effective when classes are unbalanced and is generally regarded as fairly conservative. This is due to the

fact it takes into account agreement of results by chance. The formula for this is 1-(1-percentage agreement)/(1-percentage agreement by chance). Figure 4 shows clearer the equation, where po is the percentage agreement and pe is the percentage agreement by chance.

Figure 4. Cohen Kappa Equation

$$\kappa = \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e}$$

In addition I shall comment on log loss, as this is a useful measure when assessing the confidence of results. This is because it is a likelihood measure where predictions are judged not only on if they are correct or not, but also by how likely true the algorithm has predicted it to be. For example a confident false answer will score worse than a hesitant false answer. Log loss should be minimised, with a score of 0 representing 100% confidence and 100% accuracy. The formula is -(1/number of samples)*for all combinations of examples and classes sum whether that class is correct * log of predicted probability. Figure 5 shows clearer the equation, where N is number of samples, M is number of classes, yij is whether the prediction is true or false, and pij is the predicted probability of this class.

Figure 5. Log Loss Equation

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$$

II. Analysis

Data Exploration

The data here is split into two parts, the protein expression data (77_cancer_proteomes_CPTAC_itraq.csv) and the clinical data (clinical_data_breast_cancer.csv). There are 83 Samples in the protein data set and 105 samples in the clinical data set.

Protein Expression Data

The **protein expression data** has 7994 different proteins, with values representing expression levels. The data is already normalised and in numerical form so minimum processing needs be done on it. However there is duplicate data for some patients, and only one set of expression data is kept for these patients to avoid overweighting them in the final classifier.

In addition the information in the features, ie the protein expressions, in this data are likely to be overlapping due to the nature of protein expression, proteins often form feedback loops and pathways where one up regulates another which either upregulates another or feeds back to increase expression of the first protein again. Particularly as many of the proteins come from the same or related genes, and gene information is also provided here as categorical data. The gene related fields are combined to form an index as the information could be useful to use as a reference later, however it is not used in the current models.

Another problem is the data is likely to be noisy. As mentioned above the pathways within a cell are complex and overlapping, with many different contributory factors and influences. Hormonal changes, diet, time of day etc can all be an influence and aren’t modeled here contributing noise.

To tackle this I will be using PCA to identify key differences, or information, in the data and be using these transformed features.

Clinical Data

The **clinical data** includes features that reflect clinical diagnoses or characteristics of the patient. The project will focus on the AJCC Stage.

AJCC Stages are clinical categorisations of the cancer. The stage is assigned based on the physical characteristics of the cancer, including whether it has metastasized, its presence in the lymph nodes and the size and shape of the tumour.

Metastasis is important as this is a serious progression of the cancer and prognosis. A metastasised cancer has appeared in another organ such as the liver or kidneys, meaning the cancer is no longer isolated to one organ but is spreading round the body.

Presence in the lymph nodes is also a signifier of the cancer's spread, with presence in one or more lymph nodes leading to a worsening prognosis. However, if no cancer is found in the primary lymph node this is a good sign, signifying the cancer hasn't spread to lymph nodes further afield.

Tumour size and shape is the third major characteristic in the AJCC Stages, split into four categories representing increases in size. A larger tumour signifies a more advanced cancer.

This feature includes stage I-IV, with A,B,C subdivisions of some of the stages. As I have a small unbalanced sample set I have removed the A,B and C signifiers to leave each patient with a categorisation of I-IV. This is because the subdivisions have smaller differences between them, likely caused by more subtle changes in protein expression and so harder to learn. For example Stage IIA and Stage IIB can in some cases be differentiated by whether there is no node metastasis or the very first level of node metastasis. Additionally medically these subdivisions have similar prognosis. For more information please see https://cancerstaging.org/references-tools/quickreferences/Documents/BreastMedium.pdf

In addition with the data being categorical I have used a label encoder to process it for later manipulation.

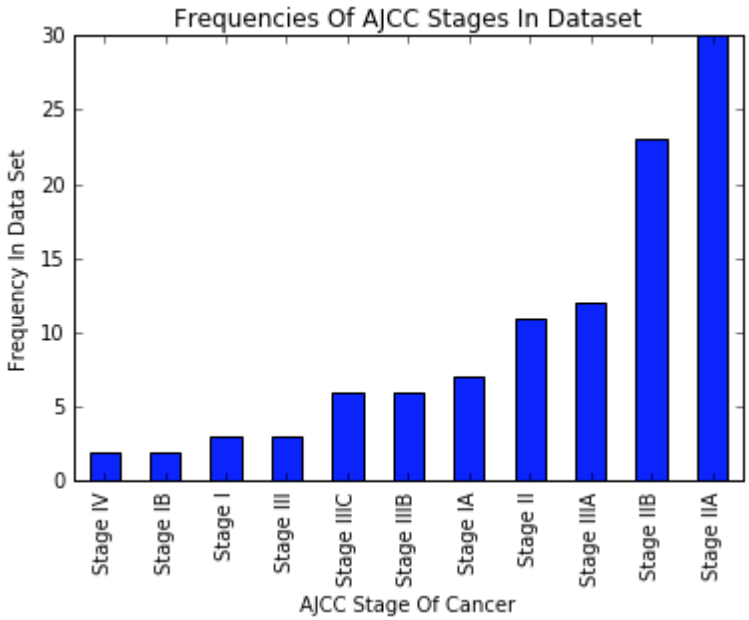
Table 1 shows the frequencies of the target variable in the total clinical data, and this reveals its unbalanced nature.

Table 1. Frequency of Subdivisions and Stages in the original Clinical Data Set

	Frequency	Frequency Without Subdivisions
Stage I	3	8
Stage IA	7	-
Stage IB	2	-
Stage II	11	49
Stage IIA	30	-
Stage IIB	23	-
Stage III	3	19
Stage IIIA	12	-
Stage IIIB	6	-
Stage IIIC	6	-
Stage IV	2	1

Exploratory Visualization

Figure 6. Column Chart of Frequencies of AJCC Stage in Dataset



Visualised in Figure 6 are the counts of the various stages in the data set, demonstrating the unbalanced nature of the categories as well as identifying likely “outliers”, categories that are poorly represented and not likely to be easy to relate to others in the set.

We can also see stage 2 is by far the most represented, with A and B variants outnumbering the less numerous stages. This means it is important to have a benchmark that at least predicts this class every time to provide a sanity check of later models performance.

Table 2. Description of Entire Protein Data Set

	count	mean	std	min	25%	50%	75%	max
count	83	83.00	83.00	83.00	83.00	83.00	83.00	83.00
mean	7994	-0.11	1.46	-11.63	-0.80	-0.05	0.65	8.44
std	0	0.20	0.15	2.25	0.21	0.12	0.13	1.70
min	7994	-0.86	1.08	-16.70	-1.87	-0.55	0.35	5.13
25%	7994	-0.22	1.36	-13.15	-0.86	-0.12	0.57	7.13
50%	7994	-0.11	1.48	-11.65	-0.76	-0.05	0.64	8.35
75%	7994	0.01	1.54	-10.23	-0.67	0.02	0.73	9.56
max	7994	0.39	1.91	-5.71	-0.52	0.23	1.14	13.22

Table 2 shows a description of all of the proteins.

For example the row min and column mean would give the value of the minimum patient value when taken over the mean of that patient's protein expressions.

We see that the values range from -24.6 to 17.6, with a mean of the protein means around 0 at -0.2. We also see the standard deviation of the mean is fairly small at 0.33, giving rise to the assumption most of proteins for most of the patients have an expression around 0.

Algorithms and Techniques

I will be using PCA (1) for feature reduction as there is likely to be overlapping information in the feature space, as well as the fact that my number of features far outweighs the number of samples. Reduction using PCA will give a reduced feature set that should give better performance when used as input for the learning algorithms later.

Principal Component Analysis (PCA) is a technique of feature reduction that looks to retain the maximum amount of information possible from each feature. It makes an assumption that the relevance of a dataset is in its variance between samples. The first principal component is the component that represents the most variance in a data set, and the next is the component that covers the next most variance, but always orthogonal to the existing chosen components so not to represent overlapping information.

This technique is appropriate here due to the likely overlapping information in the data set, and the largest differences between samples is likely to represent the most effective ways of distinguishing them. A characteristic of PCA is that it can eliminate noise as noise is likely to have less variance than the true differences, and this property is also likely to come useful here.

For feature selection I shall use Select K Best using the f_classif (2) criteria, to select the features most useful for each of the tested classifiers, hopefully giving rise to increased performance.

Select K Best simply selects the number of features specified by K, selecting those with the highest scores. Scoring in this case is based on the f_classif criteria, which is based on the ANOVA F value. This has a f distribution and is most used in analysis of variance, combining with the PCA previously performed.

For classification I shall be testing a number of learning algorithms, tuning the ones that give the best performance. These are the classification algorithms Random Forest Classifier, Support Vector Machine, Naive Bayes, Gradient Boost and Logistic Regression, in addition to one clustering algorithm, K Means. These algorithms are chosen as the represent a variety of strengths and an evaluation of them should give rise to strong candidates for tuning.

Random Forest Classifier (3) is a tree based algorithm, meaning it is strong at feature selection and isn’t affected by any possible nonlinear relationships that may be between features. It is also an ensemble algorithm that combines many weak classifiers together to deliver better performance through reducing variance. This ensemble nature should be effective here as it is likely that different samples will need to be divided using varying criteria due to the complex nature of protein data, and as overfitting is likely the reduced variance will help.

They can be slower to fit than other algorithms if number of estimators is high due to needing to train each tree. While it can be tempting to keep increasing the number of estimators performance will likely stagnate, and they are known to struggle with unbalanced data sets due to creation of biased trees.

Support Vector Machine (4) is an algorithm that simply tries to divide classes using a line as well as maximise the margin from the nearest samples to this line. It achieves this through kernels that create new features allowing these splits to be made. It is strong in that these kernels can represent almost any function so they are very

versatile, and are strong with high dimensional spaces, useful as the number of features here, even with PCA, may be similar to number of samples.

In aiming for margin maximisation they focus on splitting the data with maximum distance between the nearest samples and the splitting line. This focus on the hardest to classify examples can make them very effective, and the margin maximisation often leads to reduced overfitting as confidence is increased. Although in practice the versatility of the kernel function allows overfitting, and as well as this, often slower kernels can be used, and this also makes them slower to fit. In addition probability estimates are not directly calculated meaning computing log loss is more difficult.

Naive Bayes (5) classifies based on probabilities, with a presumption of independence between features. This presumption of independence makes Naive Bayes strong at dealing with smaller sample numbers and reducing the drain from high numbers of features (curse of dimensionality), as it need to work out less relationships. This is useful here due to the small data set. In addition because of this independence presumption they also fit quickly.

A disadvantage of this however is that the presumption of independence may not be effective here due to the likely complex, overlapping data. While Naive Bayes is often good at classifying, it performs worse at estimation.

Gradient Boost (6) is another ensemble technique, combining many weak classifiers to reduce variance and increase performance. This algorithm is particularly strong at predicting and is robust to potential outliers, useful in biological data due to relative high likelihood of outliers. However it can be easy to overfit on high numbers of estimators, and can start running much slower once number of classes increases. The algorithm that is used in this project is Extreme Gradient Boosting, which is based on the original model and is significantly quicker and more scaleable.

Logistic Regression (7) is a linear model for solving classification problems, rather than regression. This can be effective utilising probability estimates of outcomes, and is a quick algorithm to set up and run.

K Means (8) is a clustering algorithm that looks to group samples in k groups, minimizing distance of samples from the centre of its assigned cluster, minimising within cluster sum of squares. As the problem of the project effectively involves grouping samples into stages then this algorithm may prove effective. Particularly as the presumption is samples will be grouped together based on similar protein expressions for certain proteins. K Means does not deal well with high dimensionality so PCA reduction is crucial here.

Benchmark

I shall use a dummy classifier as a benchmark for my project. This shall provide a “sanity” check on my models, to ensure they are not simply picking the populous class or similar. These dummy classifiers work on simple rules, and the one used here is to predict the most frequent sample each time, as I know my set is has a very high occurrence of one class in particular. The F1 score of this dummy classifier is 0.533.

III. Methodology

Data Preprocessing

From the raw data I have inputted I have first filled NA’s in the gene symbol column as this is not going to be used in this project directly however it is good to provide reference, and leaving NA’s would throw an error.

I have then combined the gene data columns into one and set this as index, as these form descriptions of the data rather than the data itself in this project.

After identifying discrepancies in the counts of the proteins I dropped rows with NAs as this dataset has an abundance of features, and I did not want to risk my extrapolation of the values affecting the final result. The table was then transposed to have the patient identifier as the index as seen for the clinical data.

Before combining the table I also needed to make the naming conventions match, and the protein customer identifier parts were rearranged to do this.

It was then identified that some patient identifiers had multiple sets of data in the protein data and these duplicates were deleted. This was done both to avoid error from having multiple indexes as the same value, and to avoid upweighting these patients in the final model.

A lower count was then detected for the columns “Days to date of Death” in the clinical data, and as this column is unused it was removed.

Following this I combined the tables to ensure the indexes matched, and analysis revealed the protein columns had only 77 values, whereas clinical data had 105. Rows with NA were dropped to tackle this.

Due to the number of classes, their unbalanced nature and the number of samples I have combined the subclasses, denoted by A, B and C. I also removed Stage IV and Stage IB data samples, as they are they only representatives of their stages in the data I thought it likely they would represent effective outliers.

This column was then passed through the label encoder to prepare it for classification.

My data then underwent PCA to reduce the number of features and consolidate the information into a reduced set. I performed PCA to 48 features as this was the number of samples in my data set, so maximum information would be retained.

Feature reduction was then performed using Select K Best with f_classif criteria for each algorithm, with the optimum number of the PCA features then passed on for tuning.

Refer to Table 3 for the number of features chosen for each algorithm, and the F1 score on the test data (Score is F1 Score as that is the metric used for the Grid Search).

Table 3. Number of Features selected and fitted for each algorithm, with Scores from Test Set

	K Chosen	F1 Score	Accuracy	Kappa	Log Loss
Random Forest	29	0.53	0.67	0.00	3.2
Logistic Regression	4	0.55	0.59	0.00	0.87
KMeans	11	0.15	0.11	-0.12	-
Support Vector	1	0.52	0.63	-0.04	0.9
XGBoost	4	0.66	0.70	0.28	1.35
Naive Bayes	46	0.31	0.30	-0.22	1.34

Implementation

Data was split using stratified K Fold to leave out a test set of 35% of the samples. This percentage was chosen to try give a reasonable amount of samples of the smaller classes in the test set.

Initially all 6 algorithms were trained on the full feature set to give a base for future scores, and these were scored using various metrics for comparison. However F1_score was the one I focused on.

Six algorithms were implemented on the pca data, using Grid Search with Select K Best to choose the best feature set for each algorithm. The algorithms were run with default parameters at this stage for the initial

assessment to avoid additional contribution to overfitting. Grid Search was run with cv = 3 as this struck a balance between samples in the test set and numbers of iterations.

The Grid Search was also implemented using F1_score as the evaluator as this is the chosen metric on which models will be judged, with average set to weighted due to the unbalanced nature of the tests.

A function was written to perform grid search and evaluate each model, and then input results into another function that creates a scoring grid. This scoring grid includes all of the relevant metrics for classification to allow for deeper comparisons, however I have evaluated these algorithms mainly on F1_score, but also on Kappa and Log Loss.

The best features for each algorithm were then passed forward for refinement.

Refinement

The algorithms were then tuned with multiple sets of parameters. Each stage of tuning is in a separate block of code to demonstrate the progression. Each stage involves one or two parameters being tested with the Grid Search evaluate function on multiple sets of parameters. These parameters were then taken forward to the next stage. Also at each stage the scoring grid function is called to evaluate results. Though the test data grid is shown, only training data score were used to choose parameters.

For the XGBoost algorithm n_estimators was tuned first to find the correct number of estimators using a high learning rate of 0.1, with the optimum found to be 670. Then max_depth and min_child_weight were tuned as these can control large performance differences in the model. Following this gamma was tuned to provide control for overfitting, before subsample and colsample_bytree were tuned. Finally n_estimators was tuned with a lower learning rate to help control for overfitting some more, with the optimum for the new parameters (particularly the lower learning rate affects this) found to be 850.

At each stage apart from the gamma parameter (which best score was equal to the default), there was small improvements of the cross validation score. However, there was no improvement in scores on the test set, apart from for log loss which consistently came down through the tuning.

For the Random Forest algorithm I first tuned n_estimators finding the optimum to be 103, and then max features and min_samples_leaf as these are important factors in determining tree growth. Once these were tuned it was unlikely that the other parameters would give a large jump in performance, but small gains were still found by tuning min_samples_split and max_depth.

Each stage I saw improvement in the cross validation score, but once again in similarity to XGBoost, the test set scores stayed stable barring log loss.

Next I tuned Logistic Regression, specifically the parameters C and penalty, improving the cv score. I chose to use class_weight as balanced due to the knowledge my data was unbalanced.

Finally SVM was tuned by choosing the kernel, and then tuning C, before gamma was tuned for stability. Again class_weight was chosen as balanced due to the data set.

Comparison between the models revealed XGBoost was the strongest on the test set and the cross validation score on the training set overall (logistic regression did marginally better on the training set but much worse on the test), and this was kept as the final model.

Please see the appendix for final parameters of each classifier, XGBoost, Random Forest, Logistic Regression and Support Vector Machine.

Refer to Table 4 for cross validation score improvements on the training set. We have the initial scores with default parameters, followed by scores at each tuning stage. K Means and Naive Bayes weren't tuned, but are

included here for reference. Models went through differing numbers of tuning stages, hence why Logistic Regression for example only has a value for Tuning Stage 1.

Some models show a lowering score, as parameters were chosen also on knowledge of the problem, for example class weight = balanced, rather than just the data, with the hope of generalising better. This was done as the small training sample size means tuning isn't 100% accurate.

Table 4. Cross Validation Scores of Algorithms on the Training Set at each Tuning Stage

	Initial Scores	Tuning Stage 1	Tuning Stage 2
Random Forest	0.60	0.51	0.54
Logistic Regression	0.58	0.54	-
KMeans	0.42	-	-
Support Vector	0.54	0.61	0.68
XGBoost	0.58	0.64	0.67
Naive Bayes	0.51	-	-

	Tuning Stage 3	Tuning Stage 4	Tuning Stage 5
Random Forest	0.54	-	-
Logistic Regression	-	-	-
KMeans	-	-	-
Support Vector	0.68	-	-
XGBoost	0.67	0.67	0.69
Naive Bayes	-	-	-

IV. Results

Model Evaluation and Validation

The final model chosen was the XGBoost model with parameters XGBClassifier(base_score=0.5, colsample_bytree=1, gamma=0.0, learning_rate=0.05, max_delta_step=0, max_depth=3, min_child_weight=2, n_estimators=850, nthread=-1, objective='multi:softprob', seed=0, silent=True, subsample=1).

Scores were assessed using both cross validation, and an unseen data set.

These parameters are appropriate as the time to train isn't excessive and the model fits similarly on the test set and the cross validation sets implying the model generalises well and is robust.

The model performs with f1_score of 0.66 and cohen kappa score of 0.28 on the test set, which is a reasonable improvement on the dummy classifier scores of 0.533 and 0. However this isn't high enough to justify success as defined in the problem outline, as the precision and recall needed to be higher (to increase F1 score), as well as the confidence (a lower log loss). Though log loss for the model on the test set is 1.65 which is substantially better than the dummy score of 11.51. This is likely because the model gives a low probability to on its wrong predictions indicating the uncertainty, whereas the dummy classifier will always be 100% sure.

The small number of samples in the data set did cause issues with splitting the data, with creating folds meaning some of the less represented categories were split thinly.

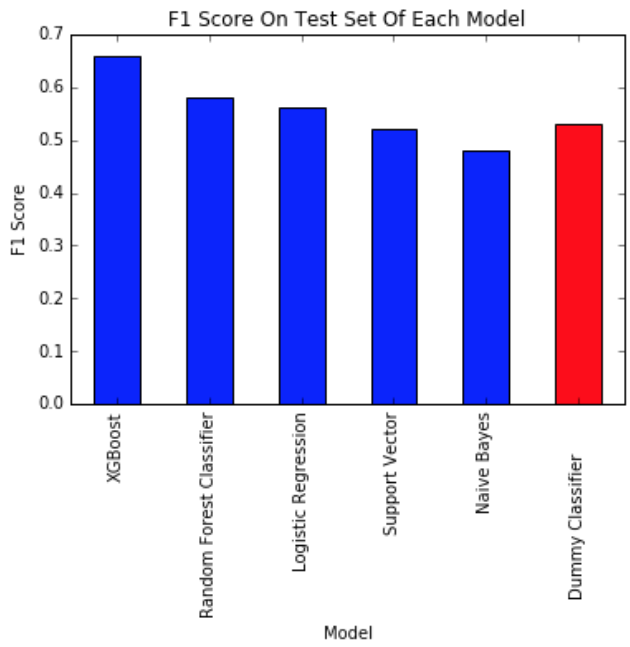
However the similarity of the cross validation score of 0.69 and test set score of 0.66 means I believe the model can be trusted to generalise, although as mentioned unfortunately the scores are too low to use in a practical sense.

Justification

Figure 7, 8, and 9 provide comparison between the final model and the dummy classifiers performance on the test set, depicting F1 Score, Log Loss and Cohen Kappa Score respectively.

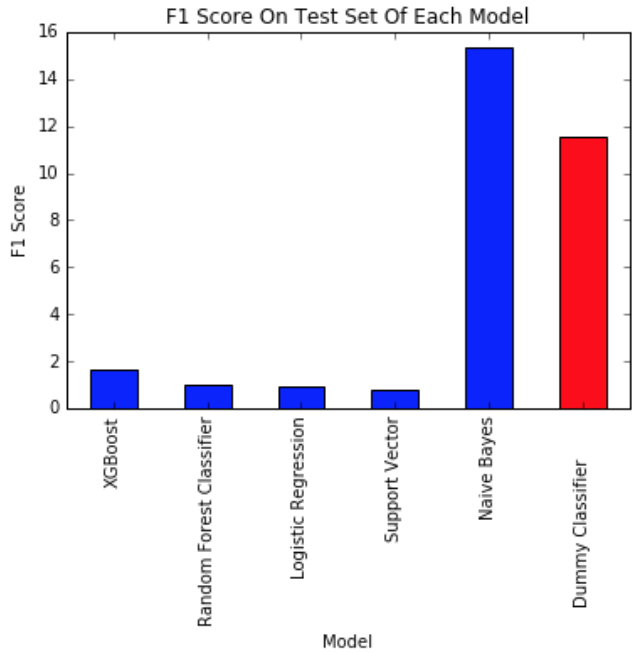
F1 Score:

Figure 7. Column Chart of F1 Score of Each Model on Test Set



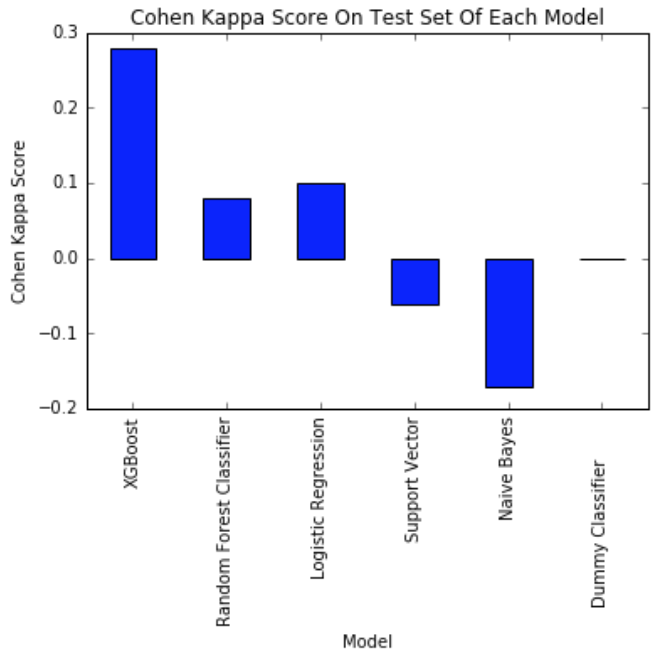
Log Loss:

Figure 8. Column Chart of Log Loss Score of Each Model on Test Set



Cohen Kappa Score

Figure 9. Column Chart of Log Loss Score of Each Model on Test Set



The final model is definitely stronger than the dummy classifier, particularly when focusing on precision showing its stronger at handling the unbalanced data than the dummy classifier, however the scores aren't strong enough to be classed as a success. In the problem outline it was stated than a high reliability of prediction was needed, and this was not achieved with the F1 Score of 0.66 on the test set.

V. Conclusion

Free-Form Visualization

Figure 10 shows the confusion matrix of the final model on the test set to demonstrate effectiveness of classifying each class. The algorithm appears to have focused too much on getting correct the most populous class, to the detriment of each of the smaller classes. This is something that could definitely be improved upon, perhaps with combining with other algorithms or through further feature processing or weighted scoring.

Figure 10. Confusion Matrix of Final Model on Test Set

Predicted	Stage 1	Stage 2	Stage 3
Actual			
Stage 1	0	2	0
Stage 2	0	16	2
Stage 3	0	4	3

Reflection

The small number of samples in the project did provide a challenge, meaning the balance between test set and how many folds of cross validation to use was important.This was made more difficult by the unbalanced classes, meaning the least represented classes were split even more thinly.

This aspect of the data set continued to affect the later processes, with PCA features being limited by the number of samples for example, meaning reduction of the protein features to the PCA features was fairly large.

The issue of the unbalanced classes I think is emphasised by the graphic above.

The final model has been found through first implementation of Principal Component Analysis on the protein data to retrieve the reduced feature set, representing the information in the form of variance from the original data. A reduced set of these was then selected using Select K Best, as chosen by criteria of ANOVA f values for each classification algorithm. This processed feature set gave performance increases on the test set, as well as the cross validation score on the training set.

A initial run was then made using these algorithms with default parameters, and the reduced feature set. After this tuning of the algorithms took place, increasing the cross validation F1 score through tuning of parameters of each model. However the test set saw more modest increases, mainly in log loss.

Finally a comparison of these models was made, before final model was chosen.

The model could use further refinement and steps that could be useful for this are detailed in the following section.

A successful solution to this problem could be useful in a clinical setting as it might allow a more automated technique of identifying cancer prognosis without relying on having the expertise of a trained doctor. In addition taking protein expressions can probably be done less invasively that the tests for determining tumor and node characteristics, which is currently needed for AJCC Stage classification. Identifying involved proteins could also lead to a better understanding of the drivers of cancer progression, and allow more targeted, progression based treatment.

Improvement

There could be significant gains in utilising the other features of the clinical data set in training, inputting more domain knowledge into the algorithm to give the model more direction.

For example the AJCC Stage takes into account metastasis, node and tumor status. Running separate models to predict each of the features, before combining these with a model predicting the final AJCC Stage would utilise this additional domain knowledge to hopefully enhance performance.

The final model in this scenario would still only take protein data as input, fulfilling this criteria.

Oversampling the underrepresented classes or undersampling the over represented classes is another technique that has had success with unbalanced data sets, giving rise to more balanced models. An example of such a technique is SMOTE, Synthetic Minority Over-sampling Technique (9).

Also running a more class weighted scoring on fitting, or layering models on top of each other may have helped tackled the issue with the focus on the most popular stage. Additionally this extra weighting could perhaps be incorporated into the feature processing.

In addition it possible that alternative criteria for KBest, as well as alternative decomposition techniques may provide a better feature set. For example Random Component Analysis has proved surprisingly effective, although I do not believe Independent Component Analysis would prove effective due to the belief that the features are largely overlapping.

I would also have liked to explore neural networks or deep learning and was unable too. However with the small dataset this would possibly have not utilised the strengths of deep learning enough.

VI. References

1. Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space" (PDF). Philosophical Magazine. 2 (11): 559–572. doi:10.1080/14786440109462720.
2. Lomax, Richard G. (2007). Statistical Concepts: A Second Course. p. 10. ISBN 0-8058-5850-4.
3. Breiman, Leo (2001). "Random Forests". Machine Learning. 45 (1): 5–32. doi:10.1023/A:1010933404324.
4. Cortes, C.; Vapnik, V. (1995). "Support-vector networks". Machine Learning. 20 (3): 273–297. doi:10.1007/BF00994018.
5. Hand, D. J.; Yu, K. (2001). "Idiot's Bayes — not so stupid after all?". International Statistical Review. 69 (3): 385–399. doi:10.2307/1403452. ISSN 0306-7734.
6. Friedman, J. H. "Greedy Function Approximation: A Gradient Boosting Machine." (February 1999)
7. David A. Freedman (2009). Statistical Models: Theory and Practice. Cambridge University Press. p. 128.
8. MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press. pp. 281–297. MR 0214227. Zbl 0214.46201
9. Chawla, N. V (2002). SMOTE: Synthetic Minority Over-sampling Technique, Journal of Artificial Intelligence Research 16 321–357

VII. Appendix

Best Parameters

XGBoost

XGBClassifier(base_score=0.5, colsample_bytree=1, gamma=0.0, learning_rate=0.05, max_delta_step=0, max_depth=3, min_child_weight=2, n_estimators=850, nthread=-1, objective='multi:softprob', seed=0, silent=True, subsample=1)

Random Forest

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini', max_depth=3, max_features=None, max_leaf_nodes=None, min_impurity_split=1e-07, min_samples_leaf=1, min_samples_split=7, min_weight_fraction_leaf=0.0, n_estimators=103, n_jobs=1, oob_score=False, random_state=0, verbose=0, warm_start=False)

Logistic Regression

LogisticRegression(C=1.0, class_weight='balanced', dual=False, fit_intercept=True, intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1, penalty='l2', random_state=0, solver='liblinear', tol=0.0001, verbose=0, warm_start=False)

Support Vector Machine

SVC(C=0.001, cache_size=200, class_weight='balanced', coef0=0.0, decision_function_shape=None, degree=3, gamma=0.1, kernel='poly', max_iter=-1, probability=True, random_state=0, shrinking=True, tol=0.001, verbose=False)