Machine Learning Engineer Nanodegree

Capstone Project

Jodie Alaine Parker

December 19th, 2016

I. Definition

Project Overview

This capstone project will focus on the Diabetic Data Set from the Pima Indian population near Phoenix, Arizona. The project will attempt to predict whether a patient has diabetes or not - this will be achieved by developing an appropriate supervised binary-classification learning system. The original owners of the data set are the National Institute of Diabetes and Kidney Diseases, who made it available publicly to the UCI Machine Learning Repository.

The UCI repository stipulates that several constraints have been placed on the selection of the instances of this data from a larger database. In particular, all patients are females and are at least 21 years old of Pima Indian heritage[1]

Diabetes mellitus (DM), commonly referred to as diabetes, is a group of metabolic diseases in which there are high blood sugar levels over a prolonged period. Diabetes is due to either the pancreas not producing enough insulin or the cells of the body not responding properly to the insulin produced. There are three types of diabetes mellitus:

- Type 1 DM results from the pancreas's failure to produce enough insulin.
- Type 2 DM begins with insulin resistance, a condition which cells fail to respond to insulin properly
- Gestational Diabetes is the their main form and occurs when pregnant women with a previous history of diabetes develop high blood sugar levels.

As of 2015, and 415 million people had diabetes worldwide, with type 2 DM making up about 90% of the cases. This represents 8.3% of the adult population, with equal rates in both women and men.[2]

Relevant papers to this dataset:

Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., & Johannes, R.S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the Symposium on Computer Applications and Medical Care} (pp. 261--265). IEEE Computer Society Press.[3]

Problem Statement

The problem statement for this capstone is to identify patients who have diabetes, based on diagnostic measurements.

This is a binary classification problem and as the data set available is labeled we can consider this a supervised task.

Strategy

1. Analyse the dataset
2. Split the dataset into training and testing sets
3. Fit the data to the most appropriate supervised classification learning algorithms.
4. Apply scoring algorithms
5. Visualise the results
6. Select the best performing systems and fine tune.
7. Select the best system and continue to tune.

Metrics

The metrics chosen are based on the fact that this is a binary classification problem domain and the dataset is unbalanced as the number of patients with a positive outcome (has diabetes) is 268 compared to the number of patients with a negative outcome(have not got diabetes) is 500. Thus, the rate of diabetes in this dataset is 34.9%. There observations can be seen in 'Figure 2.2 Description of the Dataset' in the II. Analysis section of the report.

In a binary classification task, the terms “positive” and “negative” refer to the classifier’s prediction, and the terms “true” and “false” refers to whether that prediction corresponds to the external judgement (sometimes known as the “observation”).[4]

A good metric based on the above is the F1 Score which, in statistical analysis of binary classification, is a measure of a test’s accuracy. The formula for the F1 score is:

F1 = 2 * (precision * recall) / (precision + recall)

The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst value at 0. The relative contribution of precision and recall to the F1 score are equal.[5]

Intuitively, precision is the ability of the classifier not to label as positive a sample that is negative, and recall is the ability of the classifier to find all the positive samples.

II. Analysis

Data Exploration

Information provided with the dataset from UCI Machine Learning repository:

Number of Instances: 768

Number of Attributes: 8 plus class

For Each Attribute: (all numeric-valued)

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

Missing Attribute Values: Yes

Brief statistical analysis:

Attribute Number	**Mean**	**Standard Deviation**
1	3.8	3.4
2	120.9	32.0
3	69.1	19.4
4	20.5	16.0
5	79.8	115.2
6	32.0	7.9
7	0.5	0.3
8	33.2	11.8

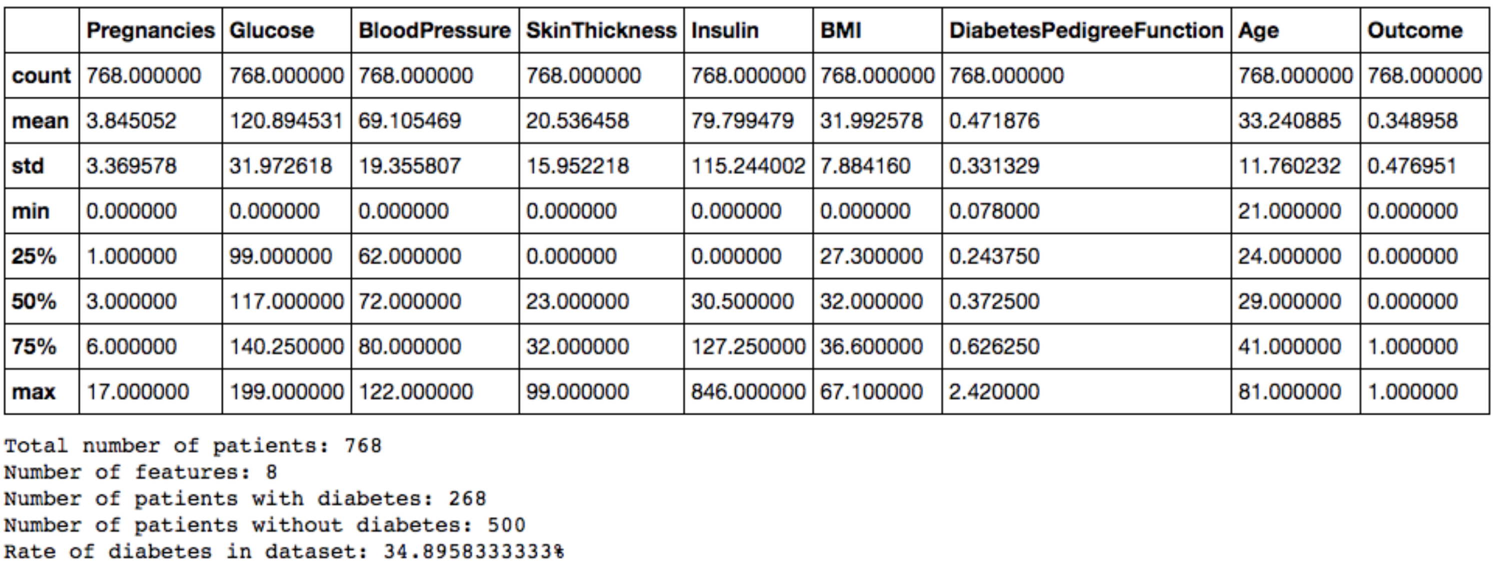
It is important to confirm the information given with an initial analysis of the dataset given and to confirm the imbalance suggested and the missing attribute values.

As we can see from 'Figure 2.1 Class Distribution' the distribution is unbalanced and supports the reasoning given for our chosen metric for our classifiers accuracy - the F1 Score.

Class Value	**Number of Instances**
0	500
1	268

Figure 2.1 Class Distribution

Furthermore, the min values shown in 'Figure 2.2 Description of the Dataset' show that we have zero values for features that it would be highly unlikely to be true. For instance Glucose, BloodPressure, SkinThickness, Insulin and BMI. We will look to process these values and replace the zero values with an average value for that feature. As it is possible for a patient not to have been pregnant we will be leaving these zero values as they are.



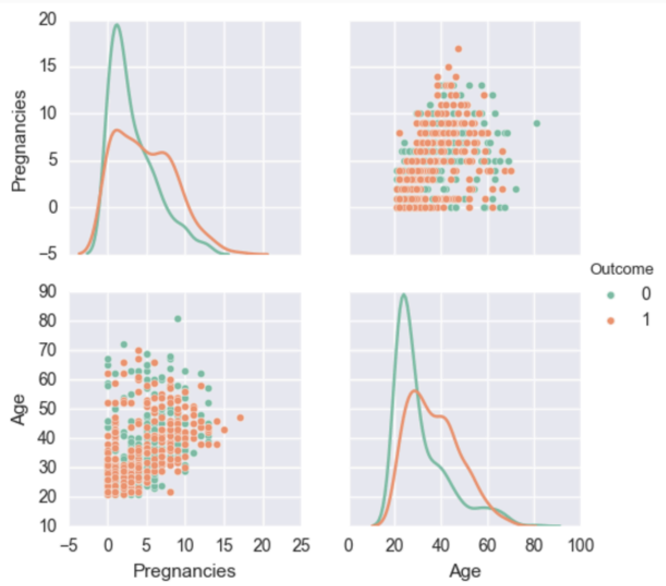
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

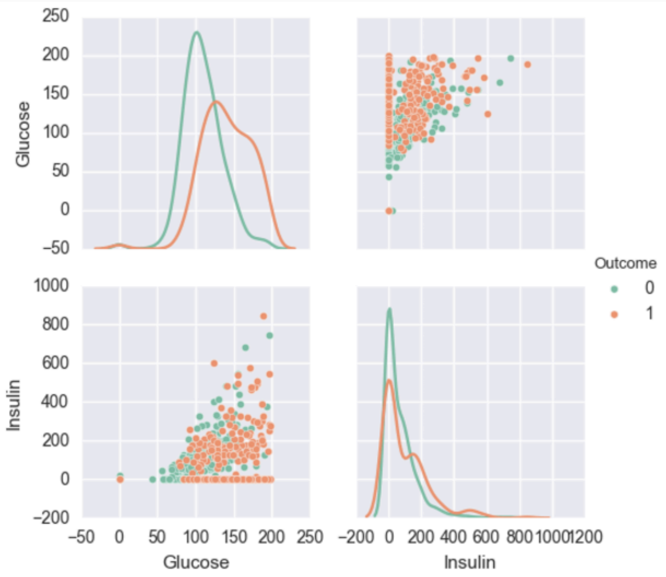
```
Total number of patients: 768
Number of features: 8
Number of patients with diabetes: 268
Number of patients without diabetes: 500
Rate of diabetes in dataset: 34.8958333333%
```

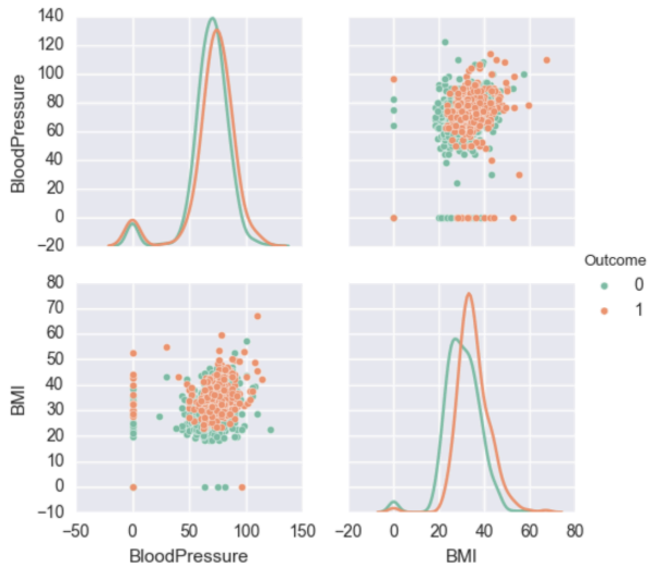
Figure 2.2 Description of the Dataset

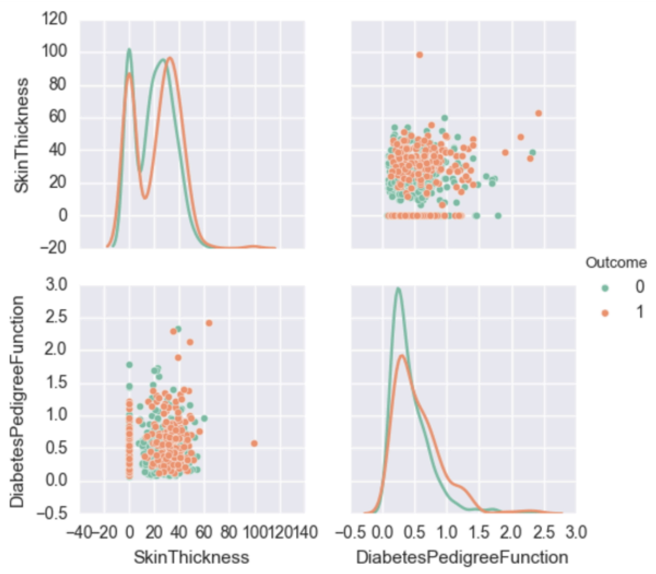
Exploratory Visualisation

A quick look at the features colour coded by 'Outcome'









A quick observation from the visualisations is that Pregnancies, Glucose, and Age seem to have an affect on whether someone has diabetes or not. Additionally there doesn't appear to be much of a correlation between the class Outcome and BloodPressure or SkinThickness.

Algorithms and Techniques

The algorithms and techniques you intend to use for solving the problem are as follows:

Techniques:

1. Randomly shuffle the data
2. Split the data into training and testing subsets
    2.1. Use 576 training points (approximately 75%)
    2.2. Use192 testing points (approximately 25%)
    2.3. Set a random_state for the functions used
    2.4. Store the results in X_train, X_test, y_train, and y_test
3. Fit a classifier to the training data
4. Make predictions using a fit classifier based on F1 score
5. Train and predict using a classifier based on F1 score

Algorithms chosen based on the problem domain has > 50 samples but < 1000 samples, it is predicting a category and has labeled data. I will complete a spot check with each of these estimators and then choose just one based on the best performance measured by the metric chosen to further tune and refine for the final classifier.

- LinearSVC(C=1.0)

This is a Linear Support Vector Classification. It is similar to SVC, included below, with parameter kernal='linear', but implemented in terms of lib linear rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples.[6]

- K-Nearest Neighbours

Neighbors-based classification is a type of instance-based learning or non-generalising learning: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbours of each point: a query point is assigned the data class which has the most representatives within the nearest neighbours the point.[7]

The optimal choice of value k for the k-neighborsClassifier is highly data-dependant: in general k surpasses the effects of noise but makes the classification boundaries less distinct.

- Random Forest Classifier

A random forest is an ensemble method - specifically part of the family of averaging methods, with a driving principle to build several estimators independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimators because its variance is reduced. The goal of such a method is to combine the predictions of several base estimators with a given learning algorithm in order to improve generalisability / robustness of a single estimator.

The random forest in particular is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control overfitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replaced if default value of True is assigned to the bootstrap parameter.[8]

- Gradient Boosting Classifier

The gradient boosting classified is a generalisation of boosting to arbitrary differentiable loss functions. It supports both binary and multi-classification. The number of week learners (i.e., regression trees) is controlled by the parameter n_estimators. The size of each tree can be controlled either by setting the tree depth via max_depth or by setting the number of leaf nodes via max_leaf_nodes. The learning_rate is a hyper-parameter in the range (0.0, 1.0] that controls overfitting via shrinkage.[9]

- SVC(random_state=2)

C-Support Vector Classification. The implementation is based on libsvm, The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to a dataset with more than a couple of 1000 samples. [6]

Benchmark

The benchmark model chosen is the largest class from the dataset, the negative result - patients that have not got diabetes. This gives us a model bench mark of 65.1%.

III. Methodology

Data Preprocessing

The following preprocessing was carried out due to their being zero values for features that it would impossible for the value to be zero:

- Preprocess the following features and convert any zero variables into average variables:
  - Blood Pressure
  - Insulin
  - Glucose
  - BMI
  - Skin Thickness

Total number of zero values in Blood Pressure feature: 35

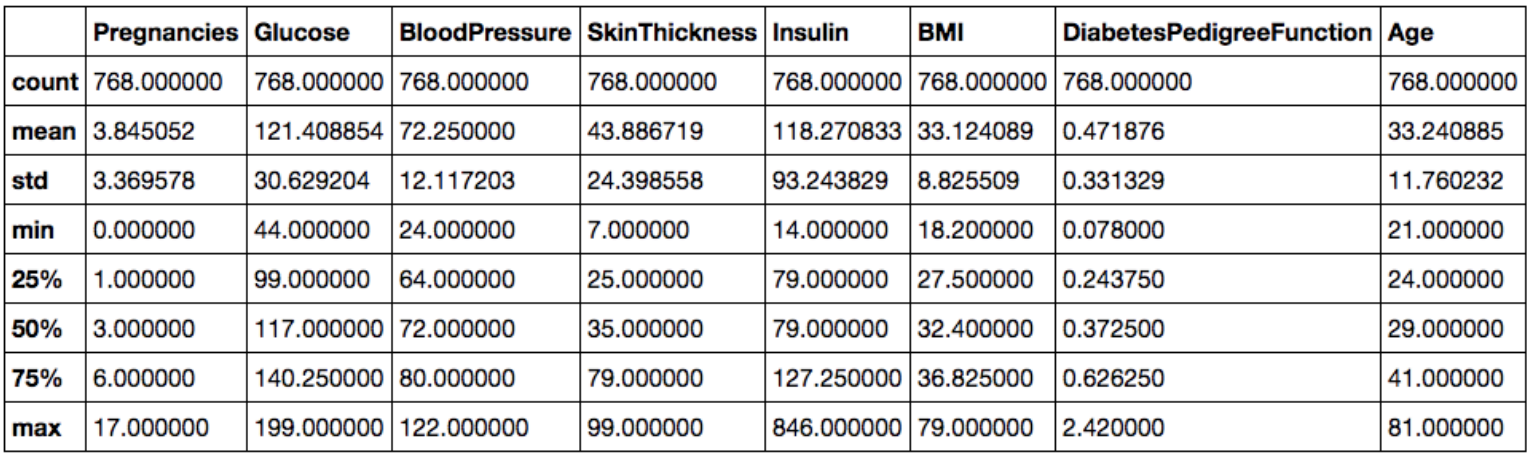
Total number of zero values in Insulin feature: 374

Total number of zero values in Glucose feature: 5

Total number of zero values in BMI: 11

Total number of zero values in Skin Thickness: 227

These zero values were observed during the data exploration and now when we describe the data the values are much more realistic, as shown in 'Figure 3.1 Description of Dataset Post Pre-Processing'.



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	121.408854	72.250000	43.886719	118.270833	33.124089	0.471876	33.240885
std	3.369578	30.629204	12.117203	24.398558	93.243829	8.825509	0.331329	11.760232
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	0.078000	21.000000
25%	1.000000	99.000000	64.000000	25.000000	79.000000	27.500000	0.243750	24.000000
50%	3.000000	117.000000	72.000000	35.000000	79.000000	32.400000	0.372500	29.000000
75%	6.000000	140.250000	80.000000	79.000000	127.250000	36.825000	0.626250	41.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	79.000000	2.420000	81.000000

Figure 3.1 Description of Dataset Post Pre-Processing

Implementation

For each of the algorithms stipulated in the Algorithms and Technique sections I completed the following:

1. Imported the learning models from sklearn
2. Initialised the for different models (LinearSVC, KNeighborsClassifier, RandomForestClassifier, and SVC)
    2.1.clf_A = LinearSVC(C=1.0)
    2.2.clf_B = KNeighborsClassifier()
    2.3.clf_C = RandomForestClassifier(n_estimators=12)
    2.4.clf_D = GradientBoostingClassifier(n_estimators=50, learning_rate=0.9, max_depth=1, random_state=0).fit(X_train, y_train)
    2.5.clf_E = SVC(random_state=2)
3. Executed the train_predict function for each classifier
4. Analysed the F1 score and prediction time taken

Refinement

Based on the analysis of the initial results of the four different classifiers the best performance based on the F1 score and prediction time taken was the GradientBoostingClassifier. The initial results on the un-tuned model were:

```
Training a GradientBoostingClassifier using a training set size of 576. . .
Trained model in 0.1301 seconds
Made predictions in 0.0039 seconds.
F1 score for training set: 0.9367.
Made predictions in 0.0009 seconds.
F1 score for test set: 0.7723.
```

Steps taken for refinement:

1. Tuning of parameters:
    1.1. loss = [‘deviance’, ‘exponential’]
    1.2. n_estimators = [30, 60]
    1.3. max_depth = [1, 2, 3, 4, 5]
    1.4. random_state = [0, 24]

2. Perform grid search on the classifier using the f1_scorer as the scoring method
3. Fit the grid search object to the training data and find the optimal parameters
4. Get the estimator
    4.1. The estimator produced:

```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
  learning_rate=0.1, loss='exponential', max_depth=3,
  max_features=None, max_leaf_nodes=None,
  min_impurity_split=1e-07, min_samples_leaf=1,
  min_samples_split=2, min_weight_fraction_leaf=0.0,
  n_estimators=60, presort='auto', random_state=0,
  subsample=1.0, verbose=0, warm_start=False)
```

The final F1 Score for the GradientBootingClassifier model has been improved, here is the tuned model score:

Training F1 score of 0.8744.

Testing F1 score of 0.7876.

IV. Results

Model Evaluation and Validation

The model chosen was the GradientBoostingClassifer which was chosen specifically for the problem domain - binary classification. This was the best performing model in the initial sample training and testing results out of the for models originally chosen. After further refinement the metric chosen for this model - F1 score - was greatly improved by tuning parameters and also making use of the grid search object.

The model completed for this project can correctly predict if a patient is diabetic or not with a score of 0.7876.

Based on the score the model can be reasonably trusted and we can see that the model is not prone to overfitting.

Justification

The final result is greatly improved from the initial results. This improvement was due to further tuning and utilisation of the grid search object.

By comparing our F1 score to the benchmark model we can be justified in the belief that the problem has been reasonably solved based on a relatively good F1 score of 0.7876. This result exceeds the benchmark model of 65.1%.

V. Conclusion

Free-Form Visualization

We can see by applying the feature_importances function, results displayed in 'Figure 4.1 Feature Importances', that Glucose, BMI, Diabetes Function and Age are considered to have the most importance. Whereas, BloodPressure and SkinThickness are considered to have the least importance. Four out of five of these importances results were originally observed in the visualisations from the 'II Analysis' section. Notably that there appeared to be a correlation between Glucose and BMI with the Outcome class, and no correlation between BloodPressure and SkinThickness with the Outcome class.

Feature	Feature Importances
Pregnancies	0.07572323
Glucose	0.22862894
BloodPressure	0.06877266
SkinThickness	0.06441205
Insulin	0.08523959
BMI	0.18972634
DiabetesPedigreeFunction	0.16520375
Age	0.12229343

Figure 4.1 Feature Importances

The relative rank (i.e. depth) of a feature used as a decision node in a tree can be used to assess the relative importance of that feature with respect to the predictability of the target variable. Features used at the top of the tree contribute to the final prediction decision of a larger fraction of the input samples. The expected fraction of the samples they contribute to can thus be used as an estimate of the relative importance of the features.[10]

Reflection

I chose this dataset for my capstone to attempt to understand a subject that was personally important to me. I wanted to focus on how much could be achieved using supervised classification algorithms.

Improvement

In order to improve this further I feel I would need to approach the problem in a completely different way. Perhaps with more time I could have looked at deep learning algorithms instead. In particular I would like to explore the MLPClassifier which implements a multi-layer perception (MLP) algorithm using back-propagation. Additionally, I would also like to further explore the additional parameters available for tuning the GradientBoostingClassifer.

References

[1] https://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes

[2] https://en.wikipedia.org/wiki/Diabetes_mellitus

[3] http://pubmedcentralcanada.ca/pmcc/articles/PMC2245318/pdf/procascamc00018-0276.pdf

[4] http://scikit-learn.org/stable/modules/model_evaluation.html#precision-recall-f-measure-metrics

[5] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

[6] http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

[7] http://scikit-learn.org/stable/modules/neighbors.html#classification

[8] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[9] http://scikit-learn.org/stable/modules/ensemble.html#gradient-boosting

[10] http://scikit-learn.org/stable/modules/ensemble.html#feature-importance-evaluation