**3. METHOD**

As mentioned earlier, we would like to classify patients into group of diabetic and non-diabetic patients based on the symptoms they are experiencing. This is defined as classification problem. Classification is a 2 steps process, a learning step and a prediction step. We split data into training data set which is used for the learning step, and test set which is used or prediction step.

We chose Decision Tree Classifier, KNN classifier, and AdaBoost classifier as described below. We later perform Hyperparameter Tuning and Cross Validation to improve the performance of our machine learning models which will be discussed in the next section.

**3.1 Decision Tree**

For our classification problem, we use Decision Tree as baseline model. Most of our attributes except for Age are binary nominal (yes or no), hence we chose Gini index as our attribute selection measure since it considers binary split for each attribute. Since the splitting point is predetermined, we calculate Gini Index to determine which attribute is used to further partition data set into its purest form.



We also discretized our numeric attribute (Age) into multiple bins as previously discussed. We later perform an experiment to leave Age as it is(discrete-valued attribute) and we would like to see how that affects that prediction power of the model. Details will be discussed in the next section.

To do this, we used scikit-learn’s decision tree classifier.

**3.2 KNN**

The second model we applied that is KNN Classifier, which is an instance-based learning algorithm that use different distance metric (Minskowski, Manhattan or Euclidean) to calculate the similarity/dissimilarity between records. There is no need for training the model. In our model, we chose Euclidean as our distance metric and number of nearest neighbors to compare to is 3. Since our attribute are asymmetric nominal valued-attribute, these is no need for normalization before using Euclidean distance.

To do this, we used scikit-learn’s KNN classifier library.

**3.3 AdaBoost**

We have been looking at one individual classifier as model to perform prediction. Another machine learning algorithm that we picked is AdaBoost Classifier, is an ensemble boosting algorithm. This can decrease variance using bagging approach (multiple classifiers to make prediction), and bias using boosting approach (the higher accuracy classifier helps the lower accuracy classifier to evaluate the misclassified records). We use 50 models of our baseline learning algorithm, decision tree classifier, to perform prediction.

We also use scikit-learn ‘s AdaBoost Classifier to do this.

**4. EXPERIMENTS AND RESULTS**

We performed four different experiments:

The first experiment, we include all the data attributes as it is (Age attribute is normalized, and number of attributes remained the same).

The second experiment, we include all the data attributes but perform Age discretization.

The third experiment, we performed attribute selection based on correlation/association level and attribute discretization. We performed Chi-square and rejected any attributes that have the level of significance > 1%.

The fourth experiment, we conducted hyperparameter tuning for decision tree and KNN. For decision tree, we focused on criterion (Ginni, Information Gain),split strategy(best, random best), max\_depth. For KNN, we focus on tuning number of neighbors. For AdaBoost, we focus on number of estimators and learning rate.

For each experiment, we used 80% of the dataset for training and withheld 20% for testing.

Our performance metric for all models was accuracy (fraction of correctly classified examples), and confusion metric to evaluate the misclassification rate, and ROC (AUC)(probability that model will be able to distinguish positive class and negative class)

4.1 Experiment 1 Result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Algorithm | Accuracy | AUC | Error\_rate |
| 0 | Decision Tree | 0.97115385 | 0.971875 | 0.02884615 |
| 1 | KNN | 0.95192308 | 0.95625 | 0.04807692 |
| 2 | AdaBoost | 0.94230769 | 0.9296875 | 0.05769231 |

4.2 Experiment 2 Result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Algorithm | Accuracy | AUC | Error\_rate |
| 0 | Decision Tree | 0.97115385 | 0.971875 | 0.02884615 |
| 1 | KNN | 0.99038462 | 0.9921875 | 0.00961538 |
| 2 | AdaBoost | 0.94230769 | 0.9390625 | 0.05769231 |

4.3 Experiment 3 Result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Algorithm | Accuracy | AUC | Error\_rate |
| 0 | Decision Tree | 0.97115385 | 0.971875 | 0.02884615 |
| 1 | KNN | 0.98076923 | 0.9796875 | 0.01923077 |
| 2 | AdaBoost | 0.90384615 | 0.89375 | 0.09615385 |

4.4 Experiment 4 Result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Algorithm | Accuracy | AUC | Error\_rate |
| 0 | Decision Tree CV | 0.98076923 | 0.9796875 | 0.01923077 |
| 1 | KNN CV | 0.99038462 | 0.9921875 | 0.00961538 |
| 2 | AdaBoost CV | 0.94230769 | 0.934375 | 0.05769231 |

Best Parameters:

Decision Tree: {'criterion': 'entropy', 'max\_depth': 9, 'splitter': 'random'}

For decision tree, the modification in terms of attributes do not change the accuracy. The best splitting attribute criterion is Information Gain, with the maximum depth of 9. The best accuracy achieved is 0.98 which was improved from 0.97.

﻿

KNN: {'algorithm': 'auto', 'metric': 'minkowski', 'n\_neighbors': 7, 'p': 2, 'weights': 'distance'}

For KNN, the best distance metric is Eucledian with the number of nearest neighbors is either 3 or 7. KNN works better with categorical attribute rather than normalized numeric attribute. Dropping uncorrelated attributes does not contribute to the prediction power of this model. Accuracy decreases from 0.99 to 0.98.

But when performing cross-validation, the accuracy improves, and remains at 0.99.

AdaBoost: {'learning\_rate': 0.9, 'n\_estimators': 14}

Though Boosting algorithm was known to yield high accuracy prediction, AdaBoost did not perform really well. The best accuracy score achieved is at 0.94. In fact, dropping columns with less correlation heavily impact the performance of the model. It tells us that even though the attributes may not directly correlated to the target class, all of them are important since it significantly contribute to the model performance in the end.

In our assessment, we will pick KNN without cross-validation as our prediction model based on the highest accuracy rate and the least error rate, and the resource-saving purpose. The best number of nearest neighbors is 3. We will not drop any attributes since all of them are important to the model performance. Any attribute that are numeric are highly recommended to be discretized before fitting into the model.