Random Forest Approach to the Practice Fusion Diabetes Classification Challenge

Jeffrey DeVince
CS-544 Health Informatics
Final Project
5/4/2015

Abstract

This paper describes a machine learning approach to the Practice Fusion Diabetes Classification Challenge. This challenge provided 10,000 de-identified electronic medical records with the goal of building a model capable of predicting the probability that a patient is diagnosed with type II diabetes. Strong emphasis was placed on the feature selection process and these features were used with a Random Forest algorithm. The approach presented in this paper achieved a log loss value of 0.38294 which would have been ranked 77th out of 146 teams in the actual competition.

1 Introduction

Diabetes is one of the largest healthcare concerns of the 21st century. Over 21 million people in the United States are diagnosed with diabetes and another 8.1 million are estimated to be undiagnosed [1]. Because of this extreme prevalence and diabetes' hazards toward health, diabetes is the seventh leading cause of death in the United States [1]. There are direct economic concerns as well since diabetes costs $245 billion per year to the United States healthcare system [2]. While the current state of diabetes is already formidable, the rate of diabetes is increasing and one of three children born in the year 2000 will develop diabetes in their lifetime [3]. Diabetes consists of type 1 diabetes and type II diabetes. 90%-95% of all diabetes cases are Type II diabetes which is preventable by making healthy food choices, participating in physical activity, and maintaining a healthy weight [4].

Because diabetes is preventable and controllable through lifestyle choices, it is important that patients are diagnosed early. One approach to improving diabetes diagnosing is to use electronic medical records to determine the most important risk factors for diabetes. These risk factors can then be used to find patients with a high risk of diabetes to ensure that they are fully evaluated. This rational was the basis for the Practice Fusion Diabetes Classification Challenge held from July to September 2012.

The Practice Fusion Diabetes Classification Challenge provided a dataset with 10,000 de-identified electronic medical records and participants aimed to build a model that best predicted the probabilities that patients were diagnosed with type II diabetes. The electronic medical records provided were vast, containing detailed patient data on allergy, diagnosis, immunization, lab test, medication, patient statistics, and many more types of data. A type II diabetes diagnosis was defined as having ICD9 codes 250, 250.0, 250.*0 or 250.*2 where the asterisk could be any number. Participants built their models based on the 10,000 electronic medical records and then were evaluated against 5,000 testing electronic medical records. The goal of this paper was to describe the development of a model for this challenge.

The Practice Fusion Diabetes dataset had several data types removed in order to make the classification challenge more difficult for the competition. The first modification was to remove patients that have diagnoses for diabetes complications without a basic diagnosis of type II diabetes. The second modification was to remove ICD9 codes 250, 250.0, 250.*0, 250.*2, 357.2, and 362.0*. The third modification was to remove diabetes medications from the medication records. The final modification was to remove lab tests that identified glucose or insulin related tests.

2 Methods

A machine learning approach using a random forest algorithm was used to develop the model presented in this paper. The first and most critical step of development was to decide on the features which would be used to train the random forest model. After feature selection, data preprocessing was performed in order to prepare the data for a random forest algorithm. Once the data was prepared, a training data set of the selected features was used to train the random forest model. This model was then applied to a testing dataset and the results of the model were analyzed and compared to other competitors' models.

2.1 Feature Selection and Preprocessing

The first features that were selected were features that have been shown to correlate with diabetes. These features were body-mass index (BMI) [5] and age [6]. BMI data was provided for each patient at each transcript from a doctor visit. This meant that each patient had multiple BMI values taken throughout the data's three year span from 2009 to 2012. To represent each patient's BMI as a single value, the median of all non-zero BMI data points was used as the BMI feature. For the patient's age, the patient's year of birth, which was provided only once for each patient, was used directly. The gender of the patient was also just provided once for each patient and was used as a feature.

At each transcript from a doctor's visit, multiple measurements in addition to BMI were often recorded. These data were very similar to the BMI data, so an identical approach was used to extract these data as features. For each data type, the median of all non-zero measurements was used as that data type's feature. The data types used were height, weight, systolic blood pressure, diastolic blood pressure, respiratory rate, heart rate, and temperature.

The last three features that were used were based on medication that the patient was taking, diagnoses of the patient, and lab tests that the patient received. Data for these three items were available in similar formats and implemented using the same method. The format of these data was that every instance, such as every time a lab test was order for that patient, was available with a link to the specific patient. This means that if a patient had a certain lab test once every six months, it may appear six times in the data. In order to account for the amount each item appeared, the feature used was the count of how many times a specific medication, diagnosis, or lab test occurred in the data.

Since there were many different types of medications, diagnoses, and lab tests in the data and each would require its own feature, it was desired to remove any items that would not be useful in training the random forest model in order to improve computation time. However, it was important that no useful data would be removed. Therefore, for each data type, specific medications, diagnoses, and lab tests were only removed if there were less than ten mentions in the 10,000 records. This process removed rare items that did not have enough instances to help prediction. By applying this removal process, the number of features was decreased from 2553 to 1478 for medication, 3902 to 2304 for diagnoses, and 340 to 219 for lab tests.

2.2 Training

A random forest model was trained using electronic medical records from 6,500 patients. The "Scikit-Learn: Machine Learning in Python" toolbox was used to provide the random forest functionality for this model [7]. The features were fit to the random forest model using 5,000 trees in the forest and the ground truth of whether or not each patient had diabetes. Ground truth was pre-defined in the dataset for this challenge based on if a patient had ICD9 codes 250, 250.0, 250.*0 or 250.*2 where the asterisk could be any number.

2.3 Testing

Once the model was trained, electronic medical records from 3447 patients were used to test the model. The goal of the model was to predict the probability between 0% and 100% that a patient had a diabetes diagnosis. The prediction from the model for each patient's data was compared to the ground truth using a log loss formula to quantify the performance of the model. The log loss was the primary quantification used to evaluate the models in this competition. Log loss is defined as: $\log loss = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y_i}) + (1 - y_i)\log(1 - \hat{y_i})$ where N is the number of patients, log is the natural logarithm, $\hat{y_i}$ is the posterior probability that the i^{th} patient has diabetes, and y_i is the ground truth where $y_i = 1$ indicates diabetes and $y_i = 0$ indicates no diabetes.

Besides the competition results, it was also very important to analyze the random forest model in order to understand what features were most used to make diabetes prediction. Each feature importance value was reported using the "feature_importances_" parameter provided from Scikit-Learn. This parameter provided a numerical value for each feature where the higher the value, the more important the feature [7].

3 Results

The random forest model achieved a log loss of 0.38294 on the testing dataset. This log loss would have earned 77th place out of 146 teams in the actual competition. Figure 1 provides a graphical comparison of this model against all competitors' best submissions. Table 1 shows the top 30 most important features and their feature importance values.

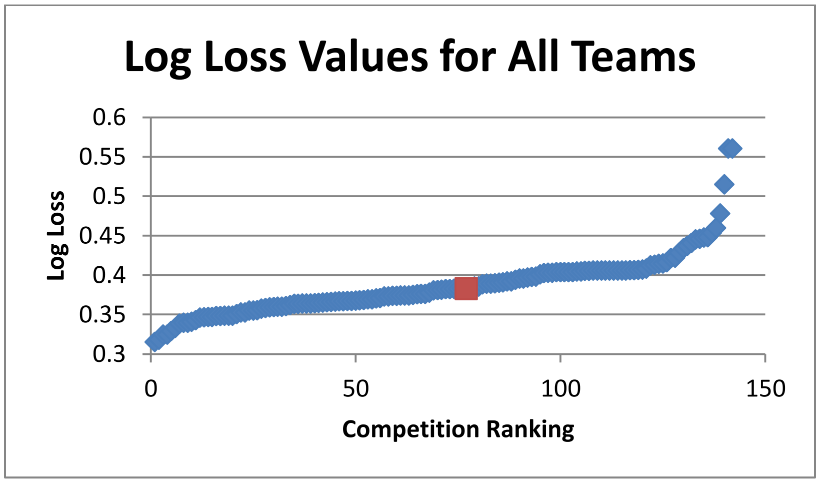

Figure 1. The log loss values for the best submission from all teams in the competition. The bottom four teams were not included because they had log loss values above 1.

Table 1. The thirty most important features used in the random forest prediction model and their feature importance values.

Data Type	Feature	Feature Importance
Patient Characteristic	Year of Birth	0.03624
Patient Characteristic	Body-Mass Index	0.03152
Patient Characteristic	Weight	0.02706
Patient Characteristic	Systolic Blood Pressure	0.02516
Patient Characteristic	Diastolic Blood Pressure	0.02166
Diagnosis	Malignant melanoma of other specified sites of skin	0.02014
Patient Characteristic	Height	0.01907
Patient Characteristic	Temperature	0.01734
Diagnosis	Anal or rectal pain	0.01601
Patient Characteristic	Respiratory Rate	0.01367
Medication	Null	0.0098
Diagnosis	Tricuspid valve disorders, specified as nonrheumatic	0.00912
Medication	Lisinopril oral tablet	0.00896
Diagnosis	Osteoarthrosis, generalized	0.00679
Diagnosis	Dysphonia	0.00636
Patient Characteristic	Gender	0.00553
Diagnosis	Coronary artery anomaly, congenital	0.0055
Medication	Simvastatin oral tablet	0.00531
Medication	Lipitor (atorvastatin) oral tablet	0.00469
Medication	Zocor (simvastatin) oral tablet	0.00455
Diagnosis	Post-traumatic headache, unspecified	0.00437
Diagnosis	Carcinoma in situ of lip, oral cavity, and pharynx	0.00431
Diagnosis	Diaphragmatic hernia without mention of obstruction or gangrene	0.00397
Diagnosis	Primary focal hyperhidrosis	0.00356
Diagnosis	Insomnia due to medical condition classified elsewhere	0.00348
Diagnosis	Dysphagia, oral phase	0.00329
Medication	Cozaar (losartan) oral tablet	0.00314
Diagnosis	Mastodynia	0.00301
Lab Test	Written Authorization	0.00293
Diagnosis	Colitis, enteritis, and gastroenteritis of presumed infectious origin	0.00282

4 Discussion

The approach to the Practice Fusion Diabetes Classification Challenge presented in this paper has shown to be comparable with other competing teams. This paper's model would have ranked about midway amongst the competing teams showing that while the model could be improved, it still performed competitively with other models.

The feature importance rankings presented in Table 1 provide an interesting insight into the random forest model presented in this paper. One key result was that the model chose body-mass index and age to be the two most important features. This was decided by the model solely based on information gain from the features and completely without any initial favoring. The fact that the model used these features as the two most important corresponds well with previous research studies showing the BMI and age are two of the largest risk factors for diabetes [5,6].

Table 1 also showed that five different medications were important features. These medications were Lisinopril, Simvastatin, Lipitor, Zocor, and Cozaar,. From the U.S. National Library of Medicine [8], the uses of these medications were analyzed and presented in Table 2.

Table 2. Description of the medications used as important features in the model.

Medication Name	Reason for Prescription	Medication Class
Lisinopril	Treat high blood pressure	Angiotensin-converting enzyme inhibitor
Simvastatin	Reduce risk of heart attack and stroke	HMG-CoA reductase inhibitor (statin)
Lipitor	Reduce risk of heart attack and stroke	HMG-CoA reductase inhibitor (statin)
Zocor	Reduce risk of heart attack and stroke	HMG-CoA reductase inhibitor (statin)
Cozaar	Treat high blood pressure	Angiotensin II receptor antagonist

All of the top five most important medications were related to heart disease. This finding by the model corresponds well to existing literature that has shown type II diabetes is associated with a higher risk of coronary heart disease [9].

While this model competitively compared to other competition submissions and corresponded well with existing literature, there are several improvements that could be made. First, the features used can be expanded to include various features such as allergies and immunizations that were available in the dataset, but not included. These features were not used because literature was not found linking these features to diabetes and it was desired to minimize the features in order to maximize computing efficiency. However, in future analysis, these features should be added in order to evaluate if they will improve the model. Another potential improvement is to combine feature in various ways and use these combinations as new features. For example, one new feature may be Weight*Age. These combinations may highlight interactions in the data, but were not used in the present model in order to minimize features and maximize computing efficiency.

Another potential future improvement is to incorporate time series into the model. In the present model, time was disregarded in order to allow for simpler analysis. This was assumed because the dataset only held data over a three year period which is a relatively short period. Also, the training data simply labeled patients as having or not having diabetes, but did not include any time granularity. Despite these assumptions, it is possible that by incorporating the time series of data into the model, the model may be improved. One theoretical example of this improvement is that if a patient has multiple specific lab tests within a short time period, this may indicate that the patient will soon be diagnosed with type II diabetes.

One caveat to the model presented in this paper is that the training data samples and testing data samples used for this model were not all the same as the actual competition. This was because the actual data samples used to evaluate the competition models were not released with diabetes labels. Because of this, the present model could not be evaluated on the same data as the models that actually participated in the competition. In order to still evaluate the model in this paper, the training dataset provided in the competition was split into a training dataset and a testing dataset for the purposes of this paper. This means that the present model was only trained on 6,500 patient records rather than 10,000 and that the model was evaluated on different and fewer patient records than the patient records used in the actual competition. Because the datasets were randomly assorted, it is still assumed that the evaluation used for this paper would lead to very similar results to the evaluation used in the actual competition. The model presented in this paper may have been disadvantaged because it was trained on 3500 fewer data samples.

5 Conclusion

The random forest model used in this paper predicted the probability of type II diabetes comparable to other models submitted to the Practice Fusion Diabetes Classification Challenge. The features that were most important in deciding the probability a patient had diabetes were found to be consistent with risk factors reported in literature. Overall, the model was a success, but room of improvement remains in order to improve the accuracy of prediction.

References

[1] Centers for Disease Control and Prevention: National diabetes statistics report: estimates of diabetes and its burden in the United States A, GA: U.S. Department of Health and Human Services; 2014.

[2] American Diabetes Association. Economic costs of diabetes in the U.S. in 2012. *Diabetes Care* April 2013. 36(4):1033-1046.

[3] Narayan KMV, Boyle JP, Thompson TJ, Sorensen SW, Williamson DF. Lifetime risk for diabetes mellitus in the United States. *JAMA* 2003 October 8;290(14):1884-1890.

[4] Centers for Disease Control and Prevention: Diabetes successes and opportunities for population-based prevention and control. U.S. Department of Health and Human Services; 2011.

[5] Mokdad AH, Ford ES, Bowman BA, Dietz WH, Vinicor F, Bales VS, Marks JS. Prevalence of obesity, diabetes, and obersity-related health risk factors. *JAMA*. 2003; 289(1):76-79.

[6] National Diabetes Information Clearinghouse. Am I at risk for type 2 diabetes? Taking Steps to Lower Your Risk of Getting Diabetes. National Institute of Diabetes and Digestive and Kidney Diseases. NIH Publication No 12-4805. June 2012.

[7] Pedregosa et al. Scikit-learn: Machine Learning in Python. JMLR. 2011; 12:2825-2830

[8] MedlinePlus: Trusted Health Information for You. United States' National Library of Medicine. National Institute of Health.

[9] Haffner S, Lehto S, Ronnemaa T, Pyorala K. Mortality from Coronary Heart Disease in Subjects with Type 2 Diabetes and in Nondiabetes subjects with and without prior Myocardial Infarction. *N Engl J Med*. 1998; 339:229-234

Appendix:

A1: Source Code

```
'''
Created on Apr 12, 2015

@author: jdevince
'''
import time
start_time = time.time()

import math
import numpy
import pickle

import sqlite3
database_loc = 'C:\\Users\\Class2015\\Documents\\Stevens
Semesters\\2015\\Spring 2015\\Health Informatics\\Final
Project\\Downloaded files\\compDataAsSQLiteDB\\compData.db'
database = sqlite3.connect(database_loc)

#X_array = list of features to be given to random forest trainer
#Y_array = list of target classes to be given to random forest trainer
X_array = []
Y_array = []


from sklearn.ensemble import RandomForestClassifier
forest_object = RandomForestClassifier(n_estimators=5000,oob_score=True)

data = database.cursor()
patientIDs = data.execute('SELECT DISTINCT PatientGuid FROM
training_patient').fetchall() #patientIDs is list of all patients
#9948 total patientIDs from training_patient

def get_all_medians_from_training_transcript(patient):
    ALL_data = data.execute("SELECT
BMI,Height,Weight,SystolicBP,DiastolicBP,RespiratoryRate,HeartRate,Tempe
rature FROM training_transcript WHERE PatientGuid= '%s'" %
patient[0]).fetchall()
    #Iterate through each feature (BMI first, then Height, then Weight...)
    #    Iterate through each row for patient (only look at one feature at
a time)
    features_list = []
    for x in range(0,8):
        THIS_values = []
        #Get a list of all none zero THIS recordings for the current patient
        for each_THIS_value in ALL_data:
            if (each_THIS_value[x] != 0.0 and each_THIS_value[x] !=
'NULL'):
                THIS_values.append(float(each_THIS_value[x]))
        if (len(THIS_values) > 0):
            """
            #Average the THIS recordings
            length = len(THIS_values)
            total_sum = 0
            for value in THIS_values:
                total_sum+=value
            average_THIS = total_sum / length
            features_list.append(average_THIS)
            """

features_list.append(numpy.median(numpy.array([THIS_values])))
        else:
            features_list.append(0)
    return features_list

def get_gender(patient):
    gender = data.execute("SELECT Gender FROM training_patient WHERE
PatientGuid= '%s'" % patient[0]).fetchone()

    if (gender[0] == 'M'):
        return 0
    elif (gender[0] == 'F'):
        return 1
    else:
        print "ERROR IN GENDER"
        return 0

def get_YearOfBirth(patient):
    YearOfBirth = data.execute("SELECT YearOfBirth FROM training_patient
WHERE PatientGuid= '%s'" % patient[0]).fetchone()[0]
    #print YearOfBirth
    return YearOfBirth

"""Remove low occurance lab tests"""
global all_tests_list

#all_tests_list = data.execute("SELECT DISTINCT HL7Text FROM
training_labs").fetchall()
##340 tests total, so 0 to 339
#
#all_tests_list.remove((None,))
#all_tests_list.remove((u'See below:',))
#all_tests_list.remove((u'.',))
#
#for each_test in all_tests_list:
#    amount = data.execute("""SELECT COUNT(HL7Text) FROM training_labs
WHERE HL7Text = "%s" """ % each_test).fetchall()
#    if (amount[0][0]<10):
#        all_tests_list.remove(each_test)
##This cuts down lab features from 340 to 219  (remove below 10)
###Above with hashtags is how all_tests_list was found. If you want a new
all_tests_list with differing conditions (not <10), then use above code.
Hardcoding like below saves time.
all_tests_list = #REMOVED FOR PRINTING

def get_HL7Text(patient):
    #Return a number indicating a count of what lab tests have been performed
for this patient
    #Example: [[0],[0],[1],[0]] if coding is
(testtype1,testtype2,testtype3,testtype4) and patient only had testtype3
    code = [0]*219
    all_tests_received = data.execute("SELECT HL7Text FROM training_labs
WHERE PatientGuid= '%s'" % patient[0]).fetchall()
    for each_test in all_tests_received:
        if (each_test in all_tests_list):
            location = all_tests_list.index(each_test)
            code[location] +=1
    return code

"""Remove low occurance DiagnosisDescription"""
global all_DiagnosisDescription_list

#all_DiagnosisDescription_list = data.execute("SELECT DISTINCT
DiagnosisDescription FROM training_diagnosis").fetchall()
##3902 types of diagnosis total
#
#x=0
#for each_diagnosis in all_DiagnosisDescription_list:
#    print x
#    x+=1
#    amount = data.execute("""SELECT COUNT(DiagnosisDescription) FROM
training_diagnosis WHERE DiagnosisDescription = "%s" """ %
each_diagnosis).fetchall()
#    if (amount[0][0]<10):
#        all_DiagnosisDescription_list.remove(each_diagnosis)
#print all_DiagnosisDescription_list
#print len(all_DiagnosisDescription_list)
#This cuts down lab features from 3902 to 2304 (below 10)
##Above with hashtags is how all_DiagnosisDescription_list was found. If you
want a new all_tests_list with differing conditions (not <10), then use above
code. Hardcoding like below saves time.

all_DiagnosisDescription_list = #REMOVED FOR PRINTING

#print all_DiagnosisDescription_list[782-299]
def get_DiagnosisDescription(patient):
    #Return a number indicating how many of each type of diagnoses have been
given for this patient
    #Example: [[0],[0],[1],[0]] if coding is
(testtype1,testtype2,testtype3 (one test),testtype4) and patient only had
testtype3
    code = [0]*2304
    all_DiagnosisDescription_received = data.execute("SELECT
DiagnosisDescription FROM training_diagnosis WHERE PatientGuid= '%s'" %
patient[0]).fetchall()
    for each_diagnosis in all_DiagnosisDescription_received:
        if (each_diagnosis in all_DiagnosisDescription_list):
            location =
all_DiagnosisDescription_list.index(each_diagnosis)
            code[location] +=1
    return code
```

```
"""MedicationName"""
global all_MedicationName_list

#all_MedicationName_list = data.execute("SELECT DISTINCT MedicationName
FROM training_allMeds").fetchall()
##2553 types of medication total
#x=0
#for each_medication in all_MedicationName_list:
#    print x
#    x+=1
#    amount = data.execute("""SELECT COUNT(MedicationName) FROM
training_medication WHERE MedicationName = "%s" """ %
each_medication).fetchall()
#    if (amount[0][0]<10):
#        all_MedicationName_list.remove(each_medication)
#print all_MedicationName_list
#print len(all_MedicationName_list)
###This cuts down lab features from 2553 to 1478 (remove below 10)
###Above with hashtags is how all_MedicationName_list was found. If you want
a new all_tests_list with differing conditions (not <10), then use above
code. Hardcoding like below saves time.


all_MedicationName_list = #REMOVED FOR PRINTING

#print all_MedicationName_list[2890-2533]
def get_MedicationName(patient):
    #Return a number indicating how many of each type of lab tests have been
performed for this patient
    #Example: [[0],[0],[1],[0]] if coding is
(testtype1,testtype2,testtype3 (one test),testtype4) and patient only had
testtype3
    code = [0]*1478
    all_MedicationName_received = data.execute("SELECT MedicationName FROM
training_medication WHERE PatientGuid= '%s'" % patient[0]).fetchall()
    for each_medication in all_MedicationName_received:
        if (each_medication in all_MedicationName_list):
            location = all_MedicationName_list.index(each_medication)
            code[location] +=1
    return code

x=0
for patient in patientIDs[0:6500]: #0 to 6500
    new_features_list = get_all_medians_from_training_transcript(patient)
#BMI through Temperature, 0-7
    new_features_list.append(get_gender(patient)) #8
    new_features_list.append(get_YearOfBirth(patient)) #9
    new_features_list = new_features_list + get_HL7Text(patient) #10-228
    new_features_list = new_features_list +
get_DiagnosisDescription(patient) #229-2532
    new_features_list = new_features_list +
get_MedicationName(patient)#2533-4010
    #print new_features_list
    X_array.append(new_features_list)
    """Target Data"""
    #Get target data (whether patient has diabetes or not)
    diabetes = data.execute("SELECT dmIndicator FROM training_patient WHERE
PatientGuid= '%s'" % patient[0]).fetchone()
    Y_array.append(bool(diabetes[0]))

    print x,
    x=x+1



forest_object = forest_object.fit(X_array, Y_array) #random forest
print
print "oob_score_ = %f" % forest_object.oob_score_
this_oob_score = forest_object.oob_score_

print "feature_importances_ = "
print forest_object.feature_importances_
print forest_object.feature_importances_.size

with open("feature_importances_", 'wb') as f:
    pickle.dump(forest_object.feature_importances_, f)
"""
for x in range(0,forest_object.feature_importances_.size):
    if forest_object.feature_importances_[x] > 0.02:
        print x,
        print forest_object.feature_importances_[x]
"""
"""Testing the VAILDATION DATA"""
correct = 0
incorrect = 0
total_tested = 0
correct_diabetes_diagnosis = 0
total_diabetes_diagnosis = 0
single_log_loss = 0.0
total_log_loss=0.0

x=0
for patient in patientIDs[6501:9948]: #6501 to 9948
    new_features_list = get_all_medians_from_training_transcript(patient)
    new_features_list.append(get_gender(patient))
    new_features_list.append(get_YearOfBirth(patient))
    new_features_list = new_features_list + get_HL7Text(patient)
    new_features_list = new_features_list +
get_DiagnosisDescription(patient)
    new_features_list = new_features_list + get_MedicationName(patient)

    predicted = forest_object.predict(new_features_list)
    proba_predicted =
forest_object.predict_proba(new_features_list)[0][1]
    correct_answer = data.execute("SELECT dmIndicator FROM
training_patient WHERE PatientGuid= '%s'" % patient[0]).fetchone()[0]
    if correct_answer == 1:
        total_diabetes_diagnosis+=1
    total_tested +=1
    if predicted == correct_answer:
        correct = correct + 1

        if predicted == 1:
            correct_diabetes_diagnosis+=1
    else:
        incorrect = incorrect + 1
    if (proba_predicted == 0):
        proba_predicted = 0.00001 #To avoid issues with taking log of zero
    if (proba_predicted == 1):
        proba_predicted = 0.99999999
    single_log_loss = 
(correct_answer*math.log(proba_predicted))+((1-correct_answer)*math.log(
1-proba_predicted))
    total_log_loss += single_log_loss
    print x,
    x+=1

print
print "Correct: %d" % correct
print "Incorrect: %d" % incorrect
print "Total Tested: %d" % total_tested
print
print "Accuracy: %f" % (float(correct)/float(total_tested))
print "Log Loss: %f" % (-total_log_loss/total_tested)
print
print "Total_Diabetes_Diagnosis: %d" % total_diabetes_diagnosis
print "Correct_Diabetes_Diagnosis: %d" % correct_diabetes_diagnosis
print
print "this_oob_score = %f" % this_oob_score
print
print "Time taken (seconds): %d" % (time.time()-start_time)
```