Prediction of Diabetes using machine learning algorithms

By: Kapil Wankhede, Nalanda Academy, Wardha

Abstract:
According to International Diabetes Federation[1], Diabetes is a chronic disease that occurs when the pancreas is no longer able to make insulin, or when the body cannot make good use of the insulin it produces. When body is not able to produce insulin or use it effectively leads to increases glucose levels in the blood (known as hyper-glycaemia). Over the long-term high glucose levels are associated with damage to the body and failure of various organs and tissues.
In this exercise our aim is to develop/build a classification model for predicting diabetes with minimum false negative. We tried various machine learning algorithms to see which one is best for this dataset. For training and building of the model we used Pima Indian Diabetic Set from University of California, Irvine (UCI) Repository.

I. INTRODUCTION:

With the development of living standards, diabetes is increasingly common in people's daily life. Therefore, it is better to detect the disease early. There have been continuous attempts by data scientists to predict diseases using various algorithms. In this data science project, we have tried to develop a machine learning model by investigating various classification algorithms.
The objective of this exercise to predict whether a patient has diabetes or not, based on certain diagnostic measurements included in the dataset.

II. DATASET:

The dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. In particular, all patients here are females at least 21 years old of Pima Indian heritage. [2]
The dataset has 9 numerical variables and has 768 patients' details.

Attributes/Features/Model Input):

1. Pregnancies: Number of times pregnant
2. Glucose: Plasma glucose concentration 2 hours in an oral glucose tolerance test
3. Blood Pressure: Diastolic blood pressure (mm Hg)
4. Skin Thickness Triceps skin fold thickness (mm)
5. Insulin: 2-Hour serum insulin (mu U/ml)
6. BMI: Body mass index (weight in kg/ (height in m) ^2)
7. Diabetes: Pedigree Function
8. Age: Age (years)

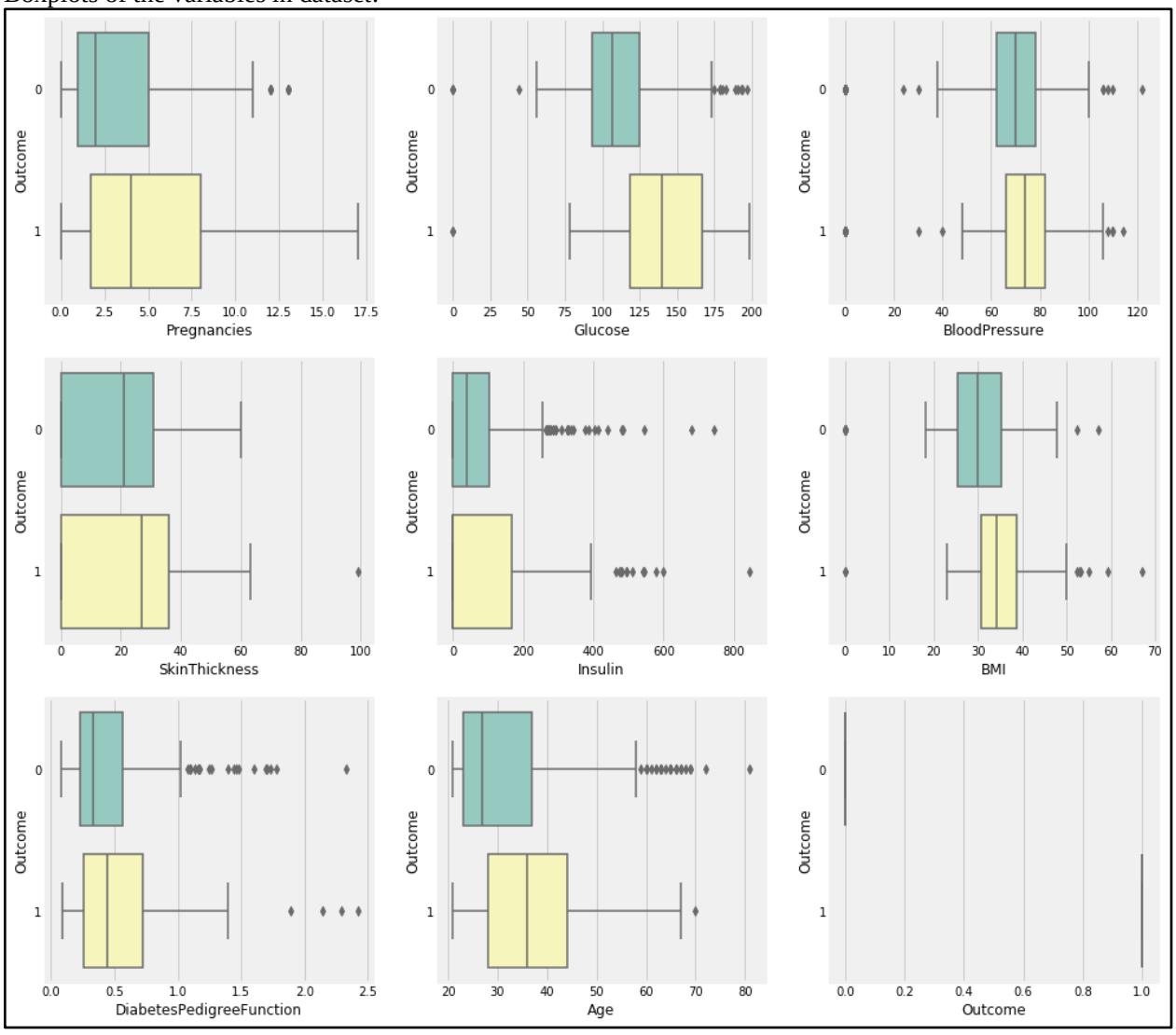
Outcome/ Target:

9. Class variable (0 :Non-Diabetic or 1: Diabetic): 268 out of 768 have Diabetes.

III. DATA PRE-PROCESSING AND MODEL TRAINING:

- There are 500 (65%) negative instances and 258 (34.9%) positive instances in the dataset.
- We created Boxplots of the dataset to see the distribution of our dataset, where we found out that:
  - There are some outliers in the dataset. We changed those outliers and Zeros with class-median of their respective variables.

Boxplots of the variables in dataset:



Correlation:

To get more insightful of the data, we plotted correlation between Target Variable and predicting variables. It shows that:

- *There is no- considerable correlation between the features and target variable. All are positively correlated with target variable.*
- *Only plasma_Glucose_con showed 0.50 correlation and all other have correlation between 0 to 0.3.*

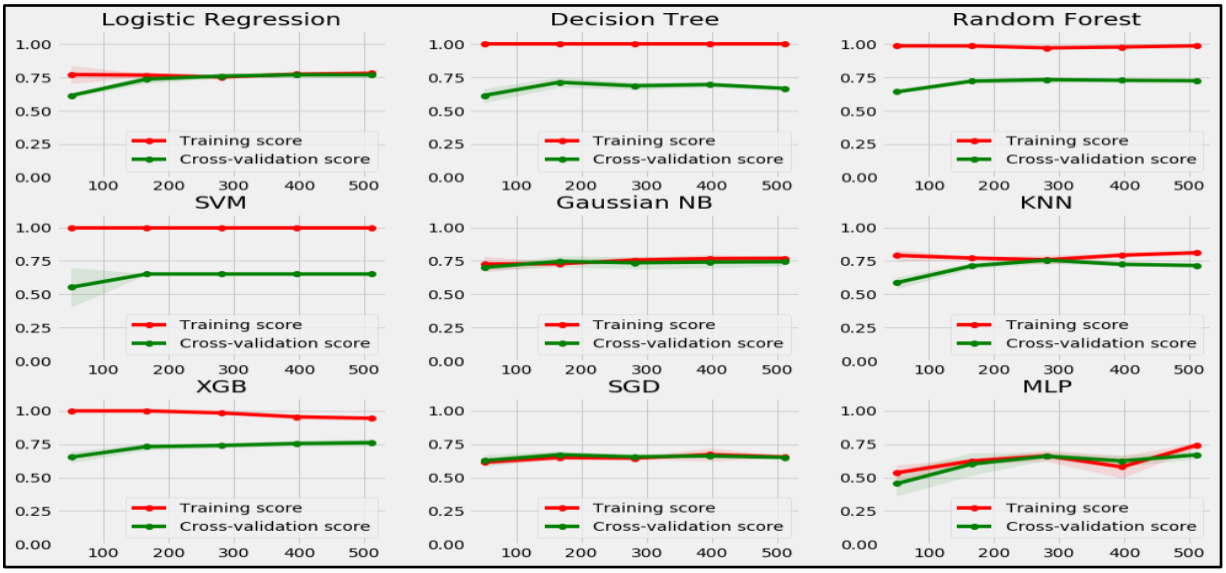
[Please check the associated Jupyeter Notebook for details]

IV. RESEARCHING THE MODEL THAT WILL BE BEST FOR THE TYPE OF DATA

Our aim is to b use first 8 variables of the dataset to predict the value of the 9th variable. Since our target is a categorical variable, we have used following classification algorithms:

1) Logistic Regression [Documentation]
2) Decision Tree [Documentation]
3) Random Forest [Documentation]
4) SVM [Documentation]
5) Gaussian NB [Documentation]
6) K- nearest Neighbors [Documentation]
7) Gradient Boost [Documentation]

We run all above ML classification algorithms to see bias-variance and other properties of the dataset. After our basic exploration, we found out that, Logistic Regression, Decision Tree, Random Forest and Gradient Boost classifier performed better than rest of the above algorithms.



V. OBSERVATIONS:

- Underfitting algorithms (Bias Issue): Logistic Regression, Gaussian NB
  - Add more features and explore complex algorithms
- Overfitting algorithms (Variance Issue): Decision Tree, Random Forest, SVM
  - Add more data
- XGB looks good here**,** if we keep adding our data samples then we might get better accuracy.

Classification Result from our first run:

	Model	Accuracy	Precision	Recall	F1	ROC	Time
1	Logistic Classifier	0.75974	0.605263	0.867925	0.713178	0.785447	0.003006
2	Decision Tree Classifier	0.805195	0.688525	0.792453	0.736842	0.802167	0.002314
3	Random Forest Classifier	0.831169	0.736842	0.792453	0.763636	0.821969	0.008955
4	SVM Classifier	0.792208	0.652174	0.849057	0.737705	0.805716	0.011528
5	Gaussian NB Classifier	0.733766	0.611111	0.622642	0.616822	0.70736	0.001601

6	KNN Classifiers	0.824675	0.724138	0.792453	0.756757	0.817018	0.005077
7	GB Classifier	0.857143	0.754098	0.867925	0.807018	0.859705	0.049313
8	XGB Classifier	0.844156	0.730159	0.867925	0.793103	0.849804	0.022139

Optimization and other performance improvement exploration:

Before finalizing optimization, we explored Feature addition and feature scaling options for the model performance improvement.

1) **Feature addition:** We added squared features into our dataset and found out, there are no significant improvement in classification performance of the algorithms. So, we moved towards our next step:
2) **Feature Scaling:** We also used feature scaling option and we did not get any better classification result than our first attempt.

Optimizations: We used grid search technique to get best parameters for our models.

Finally, we got best performance (High Accuracy and minimum false negative) with Extreme Gradient Boosting model. According to application of the model, we want a model with minimum false negative, so that we should not miss any patient with a diabetes. So, we don't have tolerance for many false positive.

```
from sklearn.model_selection import train_test_split


Best_parameters = pd.DataFrame([[0,0]],columns=['Model','Best_Parameters'])
results = pd.DataFrame([[0, 0,0,0, 0,0 ]],columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 ','ROC' ])
for i in range(10):
    X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.25)

    # Create a list of classifiers
    classifiers = {'Gradient Boosting':GradientBoostingClassifier(),
                'Exteme Gradient Boosting':XGBClassifier()}

    kfold = KFold(n_splits = 5, random_state = 123)

    # Parameters grids for that model
    models_and_parameters = {'GB':(GradientBoostingClassifier(),{'loss': ['deviance', 'exponential'],
                            'learning_rate': [.03, .1, .3, 1, 3],'n_estimators':[150,200,300],'max_depth':[2,3,4,5]}),

                            'XGB':(XGBClassifier(),{'learning_rate': [.03, .1, .3, 1, 3],
                                        'n_estimators':[50,70,100,150,200,250],'max_depth':[2,3,4,5],'scale_pos_w

                            }

    for name, (model, params) in models_and_parameters.items():
        clf = RandomizedSearchCV(estimator = model, param_distributions = params, cv = kfold, n_jobs = -1, verbose = 2,
        #print(name,":")
        #print(clf.best_params_)
        y_pred = clf.predict(X_test)
        roc = roc_auc_score(y_test, y_pred)
        acc = accuracy_score(y_test, y_pred)
        prec = precision_score(y_test, y_pred)
        rec = recall_score(y_test, y_pred)
        f1  = f1_score(y_test, y_pred)
        Bestparameters = pd.DataFrame([[name, clf.best_params_]],columns=['Model','Best_Parameters'])
        Best_parameters = Best_parameters.append(Bestparameters, ignore_index = True)
        model_results =  pd.DataFrame([[name, acc,prec,rec, f1,roc]],columns = ['Model', 'Accuracy', 'Precision', 'Reca
        results = results.append(model_results, ignore_index = True)
print("==============================================================================")
print("Gradient Boosting Classifier")
#print((results.iloc[1::2]))
print(results.iloc[1::2].describe())
print("==============================================================================")

print("Extreme Gradient Boosting Classifier" )
#print((results.iloc[2::2]))
print(results.iloc[2::2].describe())
```

Testing:

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
================================================================
Gradient Boosting Classifier
         Accuracy  Precision     Recall         F1          ROC
count   10.000000  10.000000  10.000000  10.000000  10.000000
mean     0.874479   0.869557   0.773603   0.818221   0.853456
std      0.028994   0.037242   0.059203   0.046717   0.035217
min      0.822917   0.818182   0.652174   0.725806   0.785437
25%      0.861979   0.848901   0.742411   0.800263   0.832875
50%      0.869792   0.864593   0.778571   0.813422   0.850351
75%      0.894531   0.896901   0.813517   0.845142   0.874697
max      0.921875   0.928571   0.865672   0.885496   0.908836
================================================================
Extreme Gradient Boosting Classifier
         Accuracy  Precision     Recall         F1          ROC
count   10.000000  10.000000  10.000000  10.000000  10.000000
mean     0.877083   0.834700   0.833128   0.832107   0.868335
std      0.032867   0.067367   0.055898   0.048040   0.035373
min      0.838542   0.746835   0.695652   0.755906   0.807176
25%      0.842448   0.768561   0.828237   0.791270   0.841510
50%      0.880208   0.843371   0.833403   0.839702   0.871958
75%      0.908854   0.887541   0.846790   0.875261   0.897844
max      0.916667   0.930556   0.910448   0.887417   0.915224
```

On testing dataset, we got ~81% accuracy with only 12 false negative events, which is quite good result for this exercise.

VI. CONCLUSION:

In this study, we investigate various classification algorithms to get best prediction accuracy with minimum false negative instances of diabetic patients. We explored various data-preprocessing techniques to find the best results. After all the activities we performed, we have chosen Gradient Boosting and Extreme Gradient Boosting models with tuned parameters as our final model.

Scope for further investigation:

One should try to get more that as this dataset is very small. As we have not used/explored all classification machine learning algorithms and related techniques there is a possibility to get better result. Also, we would recommend to explore deep learning algorithms to see if we can get better results.

References:

1. International Diabetes Federation, http://www.idf.org/diabetesatlas/5e/regional-overviews, (Last access date: 30th September 2012)

2. https://www.kaggle.com/uciml/pima-indians-diabetes-database