Machine Learning Engineer Nanodegree

Capstone Project

DIABETIC PREDICTION

Kunaparaju Gayatri Padma

June 28th, 2018

I. Definition

Project Overview

- In this fast-growing world, although the technology had advanced a lot there are many health problems raising at an alarming rate.
- This problem should be addressed because many people are losing their lives because they haven't realized at the earlier time.
- So, I selected one of the most common diseases diabetes. Using the data, we can identify or predict a person whether he is having diabetes are not. This might help the persons to take precautions as they can identify it earlier because it has become common and unlike past days it is not specified to only some particular age, so we can identify it earlier.
- The major motivation for choosing this project is as we have to work on real time problems that may include any field, it is major and common issue, so using this a small change can save many lives.
- **Link:** https://www.kaggle.com/uciml/pima-indians-diabetes-database

Problem Statement

- By using the dataset, we need to find whether the person has diabetes or not.
- In the dataset we have different columns which are used for the prediction of the disease.
- We need to implement machine learning techniques used for prediction of features. Using data visualizations and algorithms we need to predict the person is diabetic or not.
- In the dataset we have certain features that may cause diabetes, so we need find which features are more relevant and which are not.
- Finally it should the predict whether a person is diabetic or not for unknown dataset.

Metrics

Accuracy in classification problems is the number of correct predictions made by the model over all kinds of predictions made.

Accuracy =(TP+TN)/(TP+TN+FP+FN)

In the Numerator, are our correct predictions (True positives and True Negatives) (Marked as red in the fig above) and in the denominator, are the kind of all predictions made by the algorithm (Right as well as wrong ones).

Precision is a measure that tells us what proportion of patients that we diagnosed as having diabetes, actually had diabetes.

Precision = TP/(TP+FP)

Recall is a measure that tells us what proportion of patients that actually had cancer was diagnosed by the algorithm as having diabetes.

Recall = TP/(TP+FN)

Fβ score – measures the effectiveness of retrieval with respect to a user who attaches beta times as much importance to recall as precision.

Fβ=(1+β2)·precision·recall/(β2·precision)+recall

When beta=0.5 more emphasis is placed on precision. This is called f 0.5 score.

Here I want to use Fbeta score as main metric because as my data is not balanced, it will work better than accuracy.

II. Analysis

Data Exploration

In this section I explored the dataset and found the total records, diabetic records, non-diabetic records, what are features used.

Total number of records: 768
No of Diabetic : 268
No of non diabetic 500
Percentage of diabetic 34.8958333333%

Features Used here are:

Pregnancies: Number of times pregnant

Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test

Blood Pressure: Diastolic blood pressure (mm Hg)

Skin Thickness: Triceps skin fold thickness (mm)

Insulin: 2-Hour serum insulin (mu U/ml)

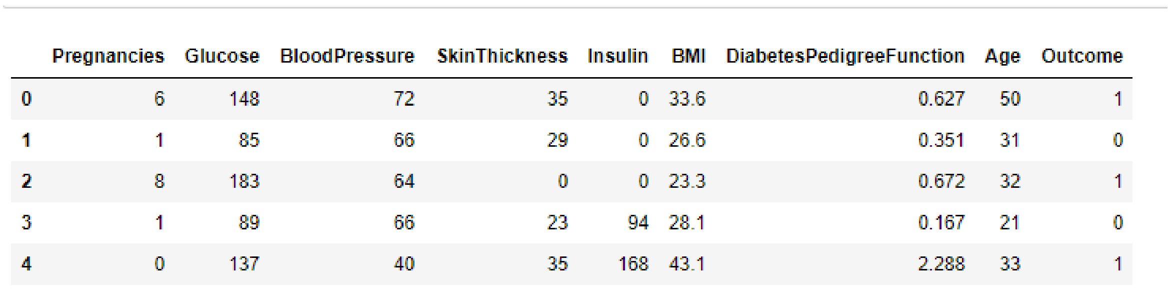
BMI: Body mass index (weight in kg/(height in m)^2)

DiabetesPedigreeFunction: Diabetes pedigree function

Age: Age (years)

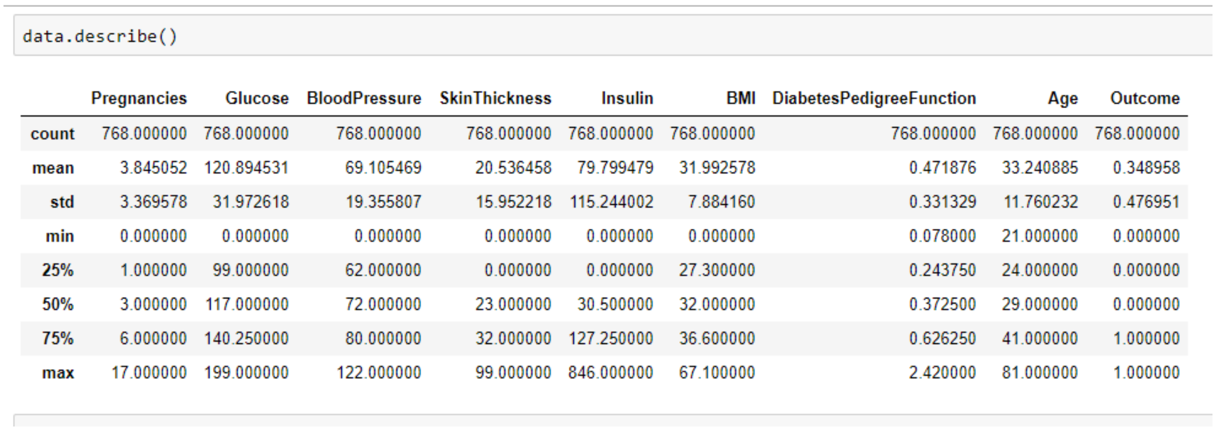
Outcome: Class variable (0 or 1)

The following figure shows the first five rows of the dataset. Here we can observe that the last column is target variable which we need to predict and other columns can be considered as features used for prediction.



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

I used one inbuilt function describe() which gives the statistical data of the given data. The following figure is output of the function.



```
data.describe()
```

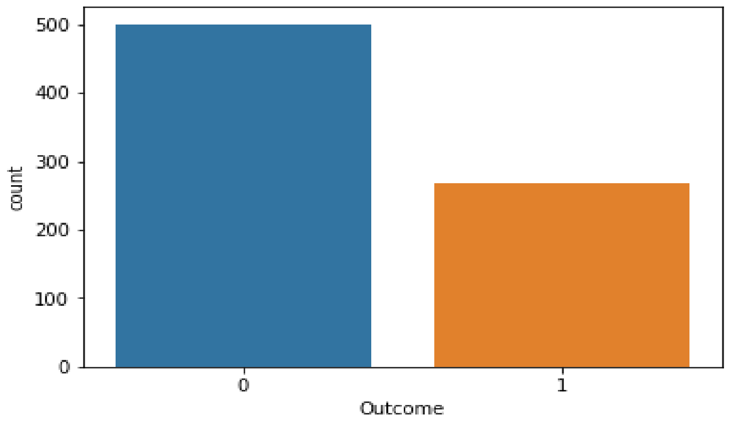
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	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

In this figure we can observe the mean, min, max, standard deviation values of every feature. By this we can identify the range of the values of features and their extreme points.

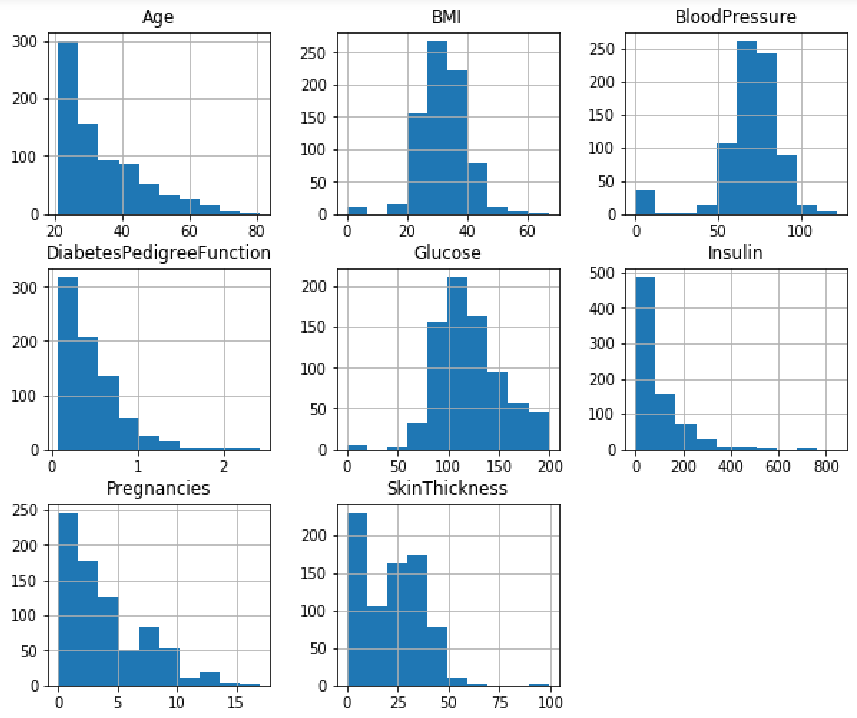
Exploratory Visualization

I have applied some visualization methods and they show how they are related and how they are varied.

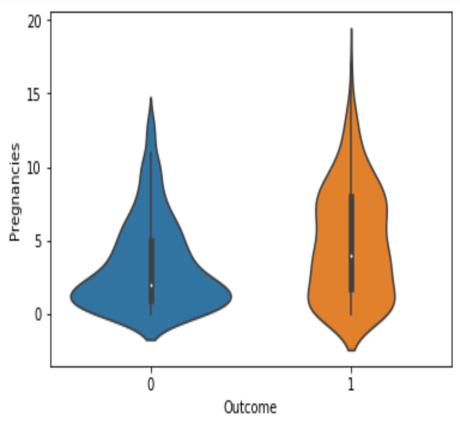
The following figure shows the Outcome variable , i.e, it shows how many are diabetic and how many are not.

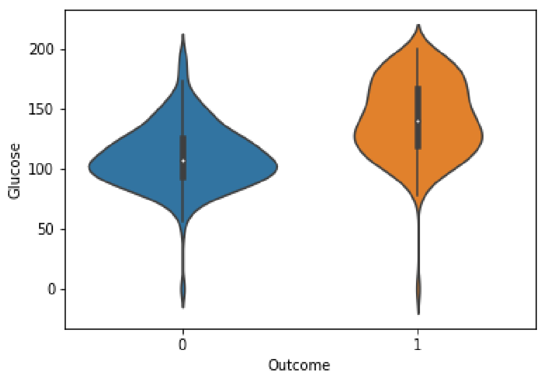


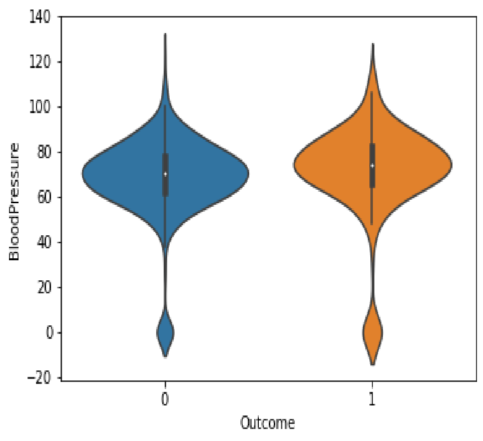
The following figure shows the histogram plots of all features and some seems like they are skewed and some seems like normalized. We can observe that Insulin have many values at the zero, this may have a chance of wrong data which we can clean in pre-processing stage.

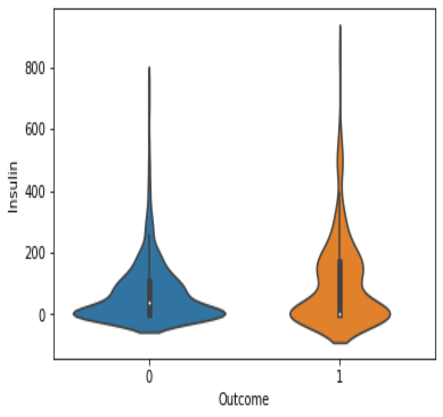


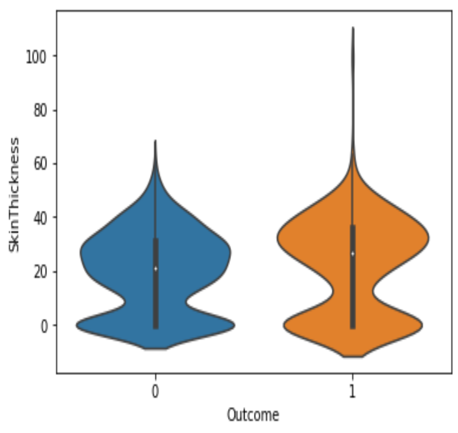
The following figures shows the violin plots of every feature with the outcome variable. I choosed violin plot because it best describes for this data than histogram and box plots.

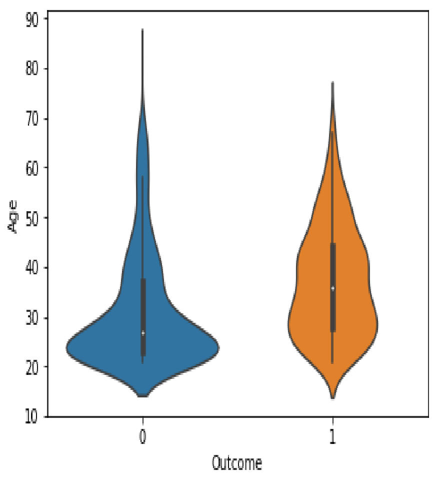




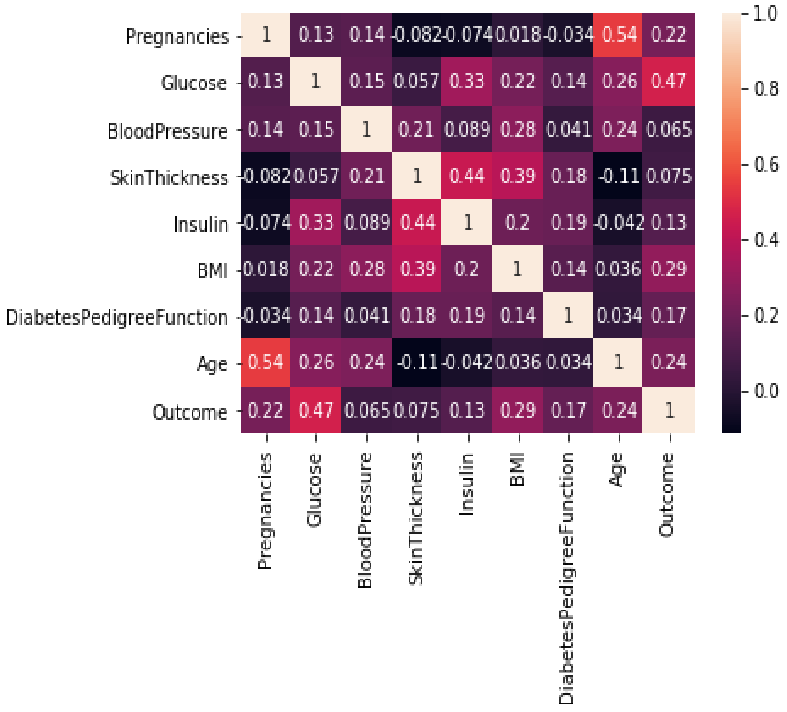








The below heat map describes how the features are correlated with each other. By observing the heat map, we can say that there is less correlation between all attributes. But every attribute is correlated with class label.



Algorithms and Techniques

As I am predicting whether the individual is diabetic or not, it means that I am classifying the two variables. So, it is binary classification. So I want to use the classification algorithms and identify which is the best.

The algorithms I used here are:

1.Logistic Regression.

2.K-Nearest Neighbours.

3.Random Forest Classifier

4.Ada Boost Classifier

1.Logistic Regression:

It classifies the data just like linear regression, but this is not regression technique. I classify the data using best fit and it is simpler algorithm. The two possible dependent variable values are often labelled as "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. In our case it is whether person is diabetic or not, so it is a binary classification and Logistic regression can be applied to this.

2.K-Nearest Neighbours:

KNearest neighbours is used for both classification and regression problems.I choose this algorithm because of its easy of interpretation and low calculation time. This algorithm is faster compared to the above algorithm. k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally, and all computation is deferred until classification.

3.RandomForestClassifier

Random forest classifier is one of the ensemble methods(where they improve results by combining models). Random forest can apply to both classification and regression. In random forest the original dataset is divided into subsets randomly and for each subset a decision tree is constructed , so many decision tree are constructed where each provides a classification.Out of all majority classification is finally considered. If there are more decision trees then more accurate the model is. It can be applied for both classification and regression. It handles missing values and also it can handle large data. It is robust to overfitting. It is more accurate for classification rather than regression.

4.AdaBoost Classifier

Adaboost(Adaptive boosting) is one of the boosting techniques. Boosting creates strong learners using weak learners. AdaBoost is one of this type where every time it creates a new learner from the past weak learners. In AdaBoost generally we use decision stamps. A decision stamp is referred as a decision tree at each level. By doing the same process repeatedly we can obtain a strong learner. As it is learned from weak learners it can give more accurate results. It is more accurate when it is binary classification. More and more decision stamps increase the memory and also increases performance time. In this project we can build decision stamps from the features and can predict accurately. Every time we create a decision stump the accuracy will be increased so that at the finding a best predictor.

Techniques:

The technique I am going to use to improve performance is GridSearchCV because tuning hyperparameters is a very important thing in improving the performance of the model. Finding best parameters is a very difficult task and gridsearchcv is a technique which takes parameters to be tuned and returns best parameter combination among all the combinations very easily.

Benchmark

Logistic regression is used as benchmark model. Fbeta score of benchmark model is reference and other model will be judge to perform better if their fbeta score will be greater than Logistic regression model. Accuracy, f-beta score and confusion matrix and will try to get better results in the ensemble learning models.

The result obtained using bench mark model is as follows:

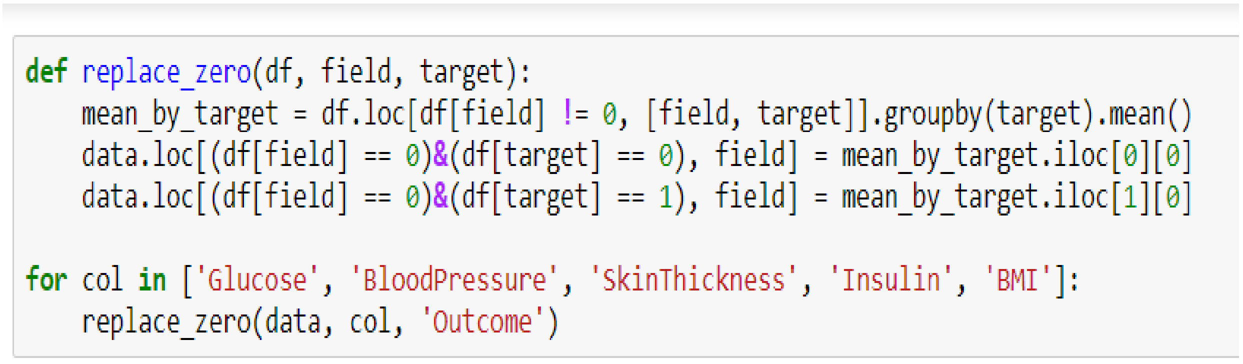
```
Accuracy:
    train: 0.809446254072
    test: 0.857142857143
F-score:
    train: 0.743440233236
    test: 0.749063670412
```

III. Methodology

Data Pre-processing

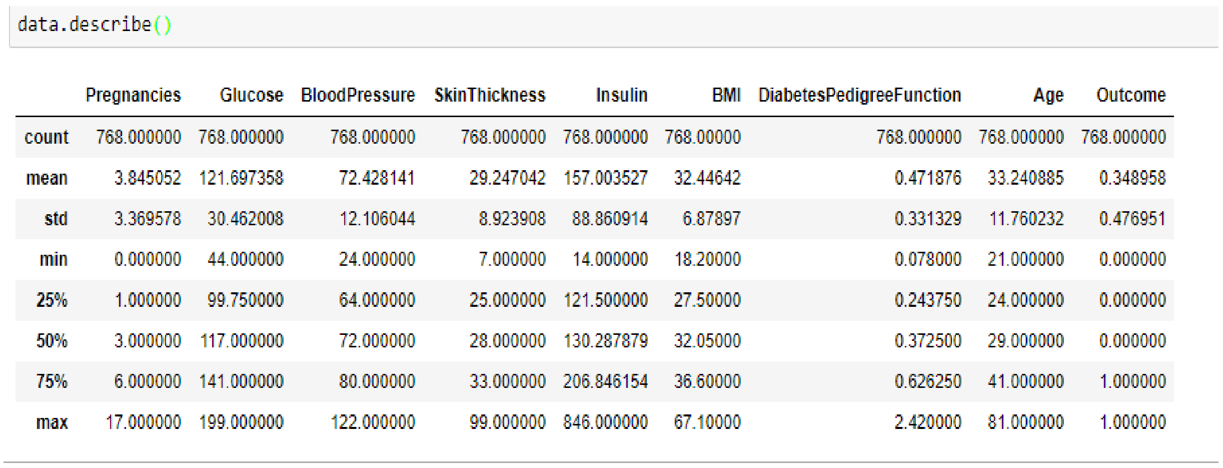
First thing I observed from the above visualizations is that some features has many zero points. This may affect the accuracy so they must be cleaned.

The below is the implementation. I replaced them with mean.



```python
def replace_zero(df, field, target):
    mean_by_target = df.loc[df[field] != 0, [field, target]].groupby(target).mean()
    data.loc[(df[field] == 0)&(df[target] == 0), field] = mean_by_target.iloc[0][0]
    data.loc[(df[field] == 0)&(df[target] == 1), field] = mean_by_target.iloc[1][0]

for col in ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']:
    replace_zero(data, col, 'Outcome')
```

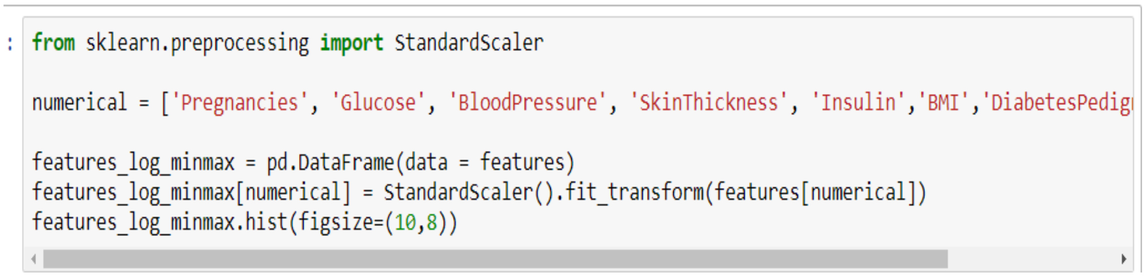
When we observe the below we can identify the difference in the above and this and all the zeros are replace with their respective means.



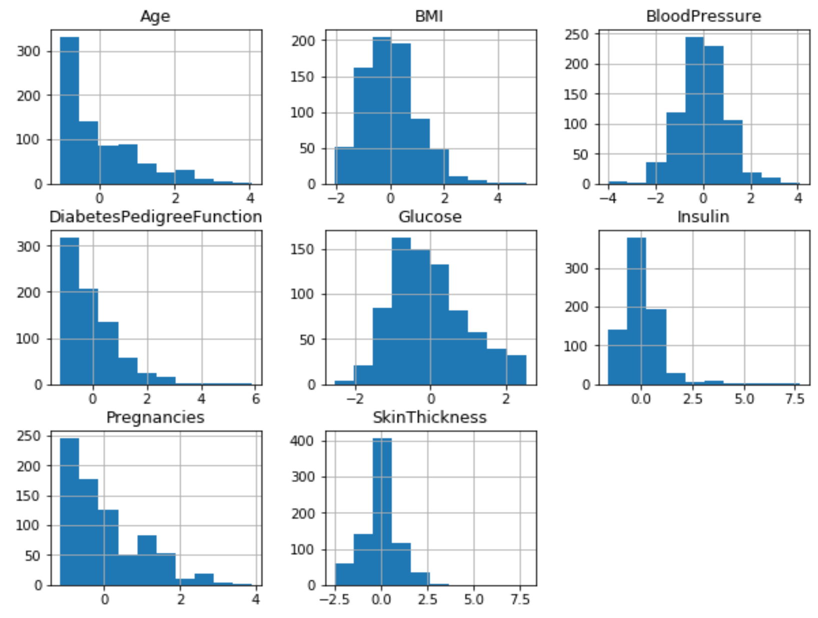
```
data.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.00000	768.000000	768.000000	768.000000
mean	3.845052	121.697358	72.428141	29.247042	157.003527	32.44642	0.471876	33.240885	0.348958
std	3.369578	30.462008	12.106044	8.923908	88.860914	6.87897	0.331329	11.760232	0.476951
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.20000	0.078000	21.000000	0.000000
25%	1.000000	99.750000	64.000000	25.000000	121.500000	27.50000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	28.000000	130.287879	32.05000	0.372500	29.000000	0.000000
75%	6.000000	141.000000	80.000000	33.000000	206.846154	36.60000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.10000	2.420000	81.000000	1.000000

Next I identified that all the features are not on one scale, as they are different attributes so Scaled them using StandardScalar.



```
: from sklearn.preprocessing import StandardScaler

  numerical = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin','BMI','DiabetesPedig

  features_log_minmax = pd.DataFrame(data = features)
  features_log_minmax[numerical] = StandardScaler().fit_transform(features[numerical])
  features_log_minmax.hist(figsize=(10,8))
```

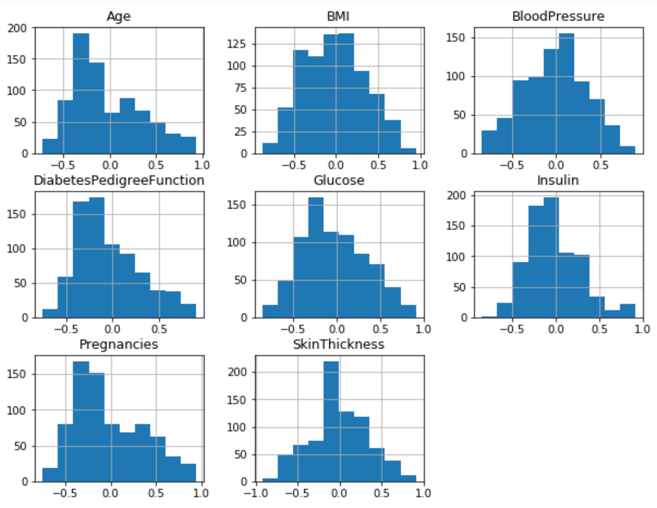


Next step is that the above graph shows that they are not in normal distribution , some are skewed so we need to normalize them in order to get better results.

So I normalized the data with normalize function.

```
: from sklearn.preprocessing import normalize

  fea_nor = pd.DataFrame(data=features_log_minmax)
  fea_nor[numerical] = normalize(features_log_minmax[numerical])

  fea_nor.hist(figsize=(10,8))
```

Comparing to above the data points are normalized now and there is skewness.



So the data is cleaned now, but we need to split the data to training and testing sets.

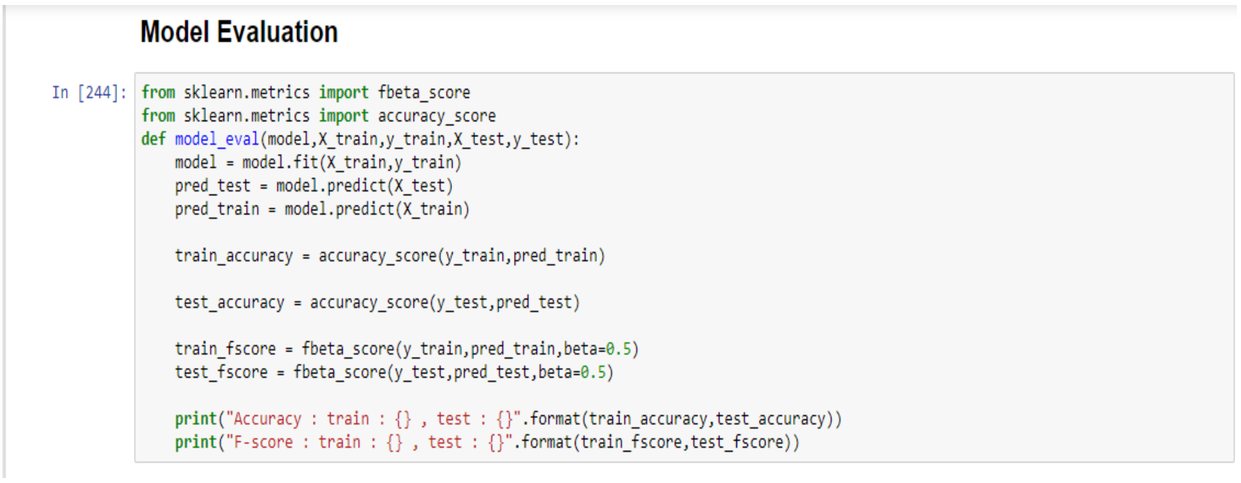
Shuffle and Split data

```
]: from sklearn.cross_validation import train_test_split

   X_train, X_test, y_train, y_test = train_test_split(fea_nor,
                                                       Outcome,
                                                       test_size = 0.2,
                                                       random_state = 0)

   print("Training set has {} samples.".format(X_train.shape[0]))
   print("Testing set has {} samples.".format(X_test.shape[0]))
```

```
Training set has 614 samples.
Testing set has 154 samples.
```

Implementation

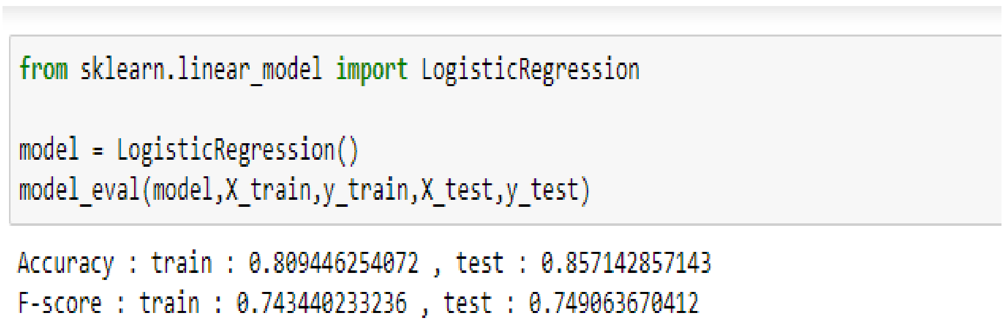
I implemented a function which is used for evaluation function, so it can used for every model we want to use. First, we need to model to the training and testing data. Next, we need to predict the values and these values can be used for finding accuracy score and f-beta score.



Model Evaluation

```
In [244]: from sklearn.metrics import fbeta_score
          from sklearn.metrics import accuracy_score
          def model_eval(model,X_train,y_train,X_test,y_test):
              model = model.fit(X_train,y_train)
              pred_test = model.predict(X_test)
              pred_train = model.predict(X_train)

              train_accuracy = accuracy_score(y_train,pred_train)

              test_accuracy = accuracy_score(y_test,pred_test)

              train_fscore = fbeta_score(y_train,pred_train,beta=0.5)
              test_fscore = fbeta_score(y_test,pred_test,beta=0.5)

              print("Accuracy : train : {} , test : {}".format(train_accuracy,test_accuracy))
              print("F-score : train : {} , test : {}".format(train_fscore,test_fscore))
```

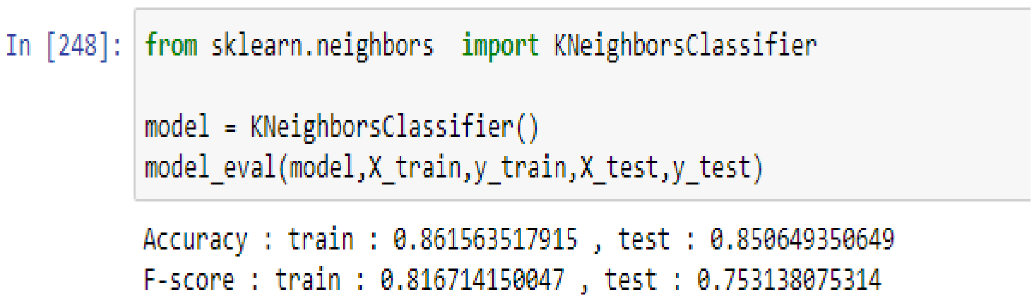
1.Logistic Regression:



```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model_eval(model,X_train,y_train,X_test,y_test)
```

```
Accuracy : train : 0.809446254072 , test : 0.857142857143
F-score : train : 0.743440233236 , test : 0.749063670412
```

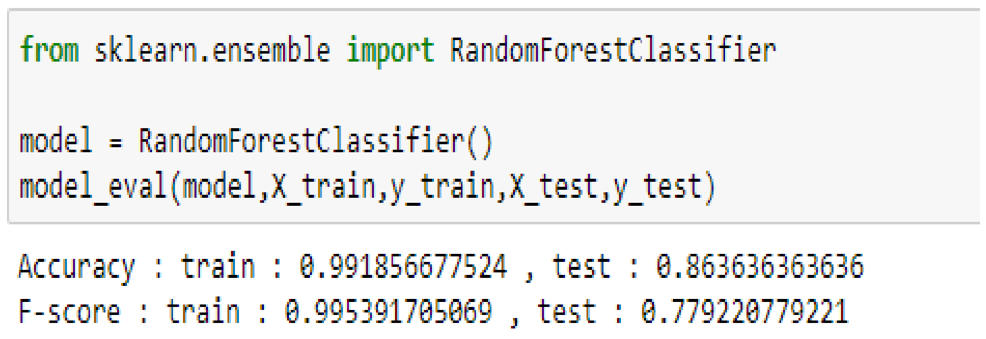
2.KNearest Neighbours:



```
In [248]: from sklearn.neighbors  import KNeighborsClassifier

          model = KNeighborsClassifier()
          model_eval(model,X_train,y_train,X_test,y_test)
```

```
Accuracy : train : 0.861563517915 , test : 0.850649350649
F-score : train : 0.816714150047 , test : 0.753138075314
```

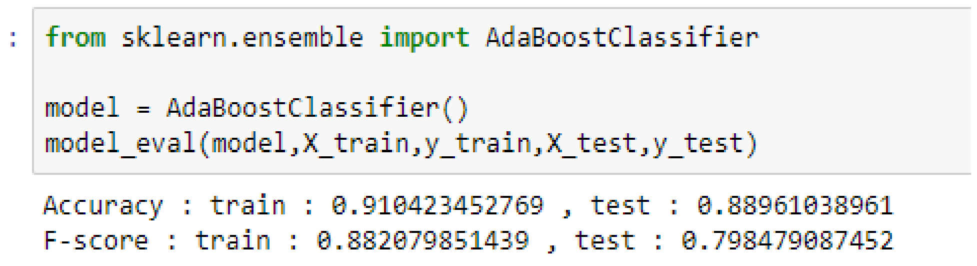
3.RandomForest:



```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()
model_eval(model,X_train,y_train,X_test,y_test)
```

```
Accuracy : train : 0.991856677524 , test : 0.863636363636
F-score : train : 0.995391705069 , test : 0.779220779221
```

4.Adaboost:



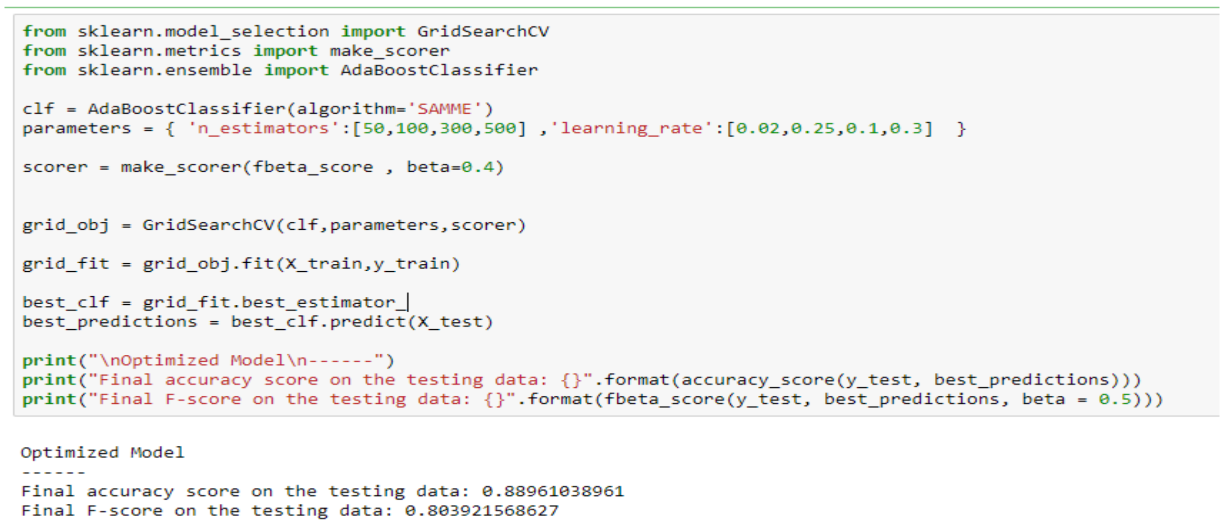
```
: from sklearn.ensemble import AdaBoostClassifier

model = AdaBoostClassifier()
model_eval(model,X_train,y_train,X_test,y_test)
```

```
Accuracy : train : 0.910423452769 , test : 0.88961038961
F-score : train : 0.882079851439 , test : 0.798479087452
```

Out of these four algorithms, I choose AdaBoost is the best classifier for this dataset. The benchmark model has a f1 score value of 74% whereas KNNclassifier has a f1 score of 75 % and for AdaBoost classifier I got 79% and I got 77% for random forest classifier. So AdaBoost is better when compared to other and when we optimized the ada boost model , got an 80% F-score.

Refinement

I used Grid search technique to improve my solution model. And the fscore is increased a little bit.



```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer
from sklearn.ensemble import AdaBoostClassifier

clf = AdaBoostClassifier(algorithm='SAMME')
parameters = { 'n_estimators':[50,100,300,500] ,'learning_rate':[0.02,0.25,0.1,0.3]  }

scorer = make_scorer(fbeta_score , beta=0.4)


grid_obj = GridSearchCV(clf,parameters,scorer)

grid_fit = grid_obj.fit(X_train,y_train)

best_clf = grid_fit.best_estimator_
best_predictions = best_clf.predict(X_test)

print("\nOptimized Model\n------")
print("Final accuracy score on the testing data: {}".format(accuracy_score(y_test, best_predictions)))
print("Final F-score on the testing data: {}".format(fbeta_score(y_test, best_predictions, beta = 0.5)))
```

```
Optimized Model
------
Final accuracy score on the testing data: 0.88961038961
Final F-score on the testing data: 0.803921568627
```

IV. Results

Model Evaluation and Validation

After the data is cleaned we need to identify which algorithms we need. As the problem is classification so we need classification algorithms. Firstly I selected the logisticregression as bench mark model as it is very simple and can be easily implemented. Then I choosed three more algorithms to compare with benchmark with my knowledge over classification algorithms which performs better.

Out of these four algorithms, I choose AdaBoost is the best classifier for this dataset. The benchmark model has a f1 score value of 74% whereas KNNclassifier has a f1 score of 75 % and for AdaBoost classifier I got 79% and I got 77% for random forest classifier. So AdaBoost is better when compared to other.

After this I tuned the hyper parameters for the adaboost classifier using gridsearch and my f1score increased by 1% which is a small value but it can be increased more if the data is more balanced.

Justification

The Logistic regression is used as benchmark model. FBeta score of benchmark model is reference and other model will be judging to perform better if their fbeta score will be greater than benchmark model. Accuracy, f-beta score will try to get better results in the ensemble learning model. With choice of different classifier in hand such as Ensemble methods and I decided to go with ensemble learning because this method increase the f-score of the models. It uses weak learner to identify the features and label it with result this was best choice. I decided to choose Ada Boost classifier as it is more accurate then others.

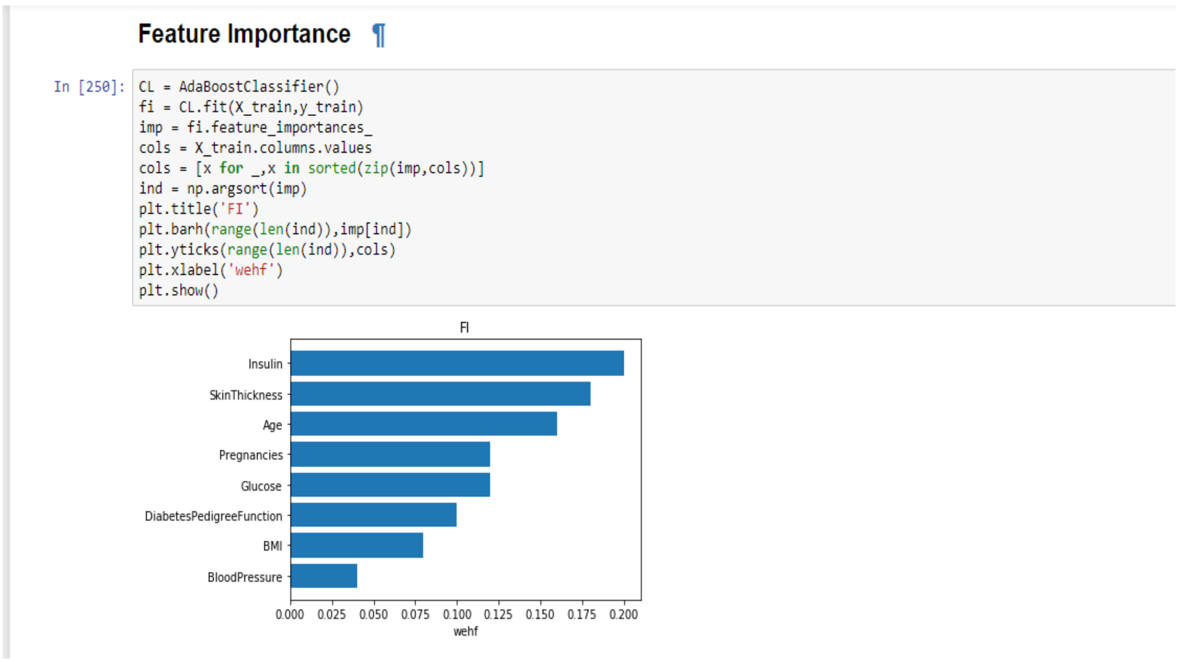
Fbeta score for benchmark model is 0.72

By seeing the F-score values of all the four models we can say that our solution model works well than the benchmark model and the other model. It’s Fscore value is high compared to remaining two models and it also takes less hyper parameters to solve the problem. Also, I tuned the selected AdaBoost model using grid search and improved fscore of 2%.

V. Conclusion

Free-Form Visualization

Using the feature importance of the ada boost classifier , I tried to identify which features are more relevant to the output. The below plot shows that Blood Pressure and BMI may be less relevant for the data when compared to other features



Feature Importance ¶

```
In [250]: CL = AdaBoostClassifier()
          fi = CL.fit(X_train,y_train)
          imp = fi.feature_importances_
          cols = X_train.columns.values
          cols = [x for _,x in sorted(zip(imp,cols))]
          ind = np.argsort(imp)
          plt.title('FI')
          plt.barh(range(len(ind)),imp[ind])
          plt.yticks(range(len(ind)),cols)
          plt.xlabel('wehf')
          plt.show()
```

Reflection

1.First, I selected my dataset from Kaggle website and imported necessary libraries and read the dataset into a pandas dataframe.

2.Then I defined my problem statement i.e., domain knowledge and to predict diabetes using the features.

3.After I described my data using describe () and have shown no of instances of each attribute in my data.

4.I visualized each and every feature with Outcome and shown they are related.

5.Then I plotted heat map for my data set to know the correlation between each attribute.

6.Then I cleaned the data by eliminating zeros and standardized the model using scaling and normalization.

7.Then I selected four different models for my problem and selected

LogisticRegression as my benchmark model and I got less performance.

8.Then I compared the performance of my bench mark model with other models. AdaBoost classifier got high f-score when compared to other models.

9.Then I performed Parameter tuning using Grid Search CV for Ada BoostClassifier and my f-score value has increased 2%.

10.I find it difficult when I am using Gridsearch technique because I already got nearly accurate result I thought there would be no improvement but then I found my f-score has increased 2%.

11.Then I applied feature importance i.e, I identified which features are more relevant and which are not.

Improvement

1.I think the results can be improved if I have more balanced data, then automatically the performance of models will increase after parameter tuning.

2.I think using SVM model or Gradient Descend or other ensemble methods also we can achieve better results which is also same as Adaboost.

3.I don’t think that I can get better results If I used my final solution as my new benchmark model because the data is highly unbalanced.