STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY

Prediction of Diabetes using Health Indicators

Big Data Technology (BIA-678-A) Fall 2021

Final Report

Instructor: David Belanger

Abdulrazzak Moulvi, Deepanshu Negi, Rishi Singh

Team 1

Abstract

After the emergence of the COVID-19 outbreak, health has become a priority for many individuals. Diabetes is one of the diseases that has been on the rise for the past 12-15 years. During the COVID-19 pandemic, it was also noted that diabetic people were particularly vulnerable. Because of its ease of onset and difficulties of complete eradication, diabetes is frequently addressed. We will use PySpark to understand the diabetes health indicator data set and conduct an elaborate Exploratory Data Analysis (EDA). We also aim to use three machine learning algorithms to construct a binary classification that can predict whether or not a person has diabetes. Finally, the outcomes are evaluated in terms of performance and scalability.

Introduction

What is Diabetes, and how does it affect you? Diabetes is a life-threatening disorder in which a person's blood glucose levels are too high. It can occur when his or her body does not create enough insulin, the insulin produced is ineffective, or the body is unable to produce insulin at all. There are several forms of diabetes, the two most common of which are type 1 and type 2. Type 1 diabetes occurs when a person's body produces no insulin at all. Type 2 diabetes occurs when the insulin produced by your body is either ineffective or insufficient. Diabetes is one of the most common chronic diseases in the United States, affecting millions of people each year and costing the economy a large amount of money. Diabetes is a significant chronic disease that can diminish one's quality of life and shorten one's lifespan. Predictive models for diabetes risk are crucial tools for public and public health professionals since early diagnosis can lead to lifestyle changes and more effective treatment. It's also crucial to recognize the scope of the problem. According to the Centers for Disease Control and Prevention, 34.2 million Americans have diabetes and 88

million have prediabetes as of 2018 [1]. Furthermore, according to the CDC, one out of every five diabetics and about eight out of ten pre-diabetics are unaware of their risk. A large portion of the disease's burden is borne by people of lower socioeconomic standing. Diabetes also has a significant economic impact, with annual expenditures for diagnosed diabetes nearing $327 billion dollars and total costs for undiagnosed diabetes and prediabetes approaching $400 billion dollars [2].

Diabetes has more than doubled in the twenty-first century. Diabetes is one of the most deadly and chronic diseases, causing blood sugar levels to rise and potentially death. Diabetes is increasing at an alarming rate, and it's especially crucial to note Type 2 diabetes in children and young adults. In the United Kingdom, type 2 diabetes affects over 90% of all individuals. In the United States, nearly one out of every ten persons has diabetes, and one out of every three adults has prediabetes. You have a blood sugar level that is greater than normal if you have pre-diabetes. It isn't high enough to qualify as type 2. According to early research, 25% of those hospitalized with the COVID-19 virus had diabetes, implying that COVID-19 had a greater impact on persons with diabetes [3]. The recovery rate of covid-affected diabetes patients was also shown to be relatively poor. As a result, diabetics are more prone to suffer major problems and die because of the virus.

In this paper we aim to create a machine learning model using PySpark to predict if patients have diabetes or not using the diabetes health indicators dataset found on Kaggle. We aim to explore the dataset and perform necessary cleaning, feature selection and normalization. The paper also aims to implement three machine learning models on imbalanced dataset and a balanced dataset

created using the Synthetic Minority Oversampling Technique. These algorithms are then to be compared and evaluated using AWS EMR.

Data Understanding

The data set was found on Kaggle and consisted of 22 columns and 253680 records a schema of these columns can be seen in figure 1. The diabetes column is divided into three groups: 0 indicates no diabetes, 1 indicates pre-diabetes, and 2 indicates diabetes. Our aim of this project is to predict whether an individual has diabetes or not hence the team decided to remove the pre-diabetes records and replace the 2 value data that indicates diabetes with a value of 1 making the model a binary classification. The following columns, such as blood pressure, cholesterol level, cholesterol check, BMI, and so on, are quite important for our prognosis. As you can see, the majority of these columns are divided into two categories: zeros and ones. In the blood pressure column, a value of zero indicates that the person does not have high blood pressure, while a value of one indicates that the person does have high blood pressure. In the next column, zero denotes the absence of high cholesterol and one denotes the existence of high cholesterol. In the next column, zero means that the cholesterol level has not been examined in the last five years, whereas one indicates that the cholesterol level has been checked in the previous five years. Except for the Body Mass Index, which was in standard form and we kept it in that form, this data was extremely pleasant because most of these variables were already in zeros and ones. Similarly, the other columns all have their own different indications - whether an individual is a smoker or not, if he or she has had a stroke, whether he or she has had a heart attack, and so on, all of which are represented by a zero or a one. Due to the

```
root
 |-- Diabetes_012: double (nullable = true)
 |-- HighBP: double (nullable = true)
 |-- HighChol: double (nullable = true)
 |-- CholCheck: double (nullable = true)
 |-- BMI: double (nullable = true)
 |-- Smoker: double (nullable = true)
 |-- Stroke: double (nullable = true)
 |-- HeartDiseaseorAttack: double (nullable = true)
 |-- PhysActivity: double (nullable = true)
 |-- Fruits: double (nullable = true)
 |-- Veggies: double (nullable = true)
 |-- HvyAlcoholConsump: double (nullable = true)
 |-- AnyHealthcare: double (nullable = true)
 |-- NoDocbcCost: double (nullable = true)
 |-- GenHlth: double (nullable = true)
 |-- MentHlth: double (nullable = true)
 |-- PhysHlth: double (nullable = true)
 |-- DiffWalk: double (nullable = true)
 |-- Sex: double (nullable = true)
 |-- Age: double (nullable = true)
 |-- Education: double (nullable = true)
 |-- Income: double (nullable = true)
```

Figure 1: Data Understanding of Columns

dataset being cleaned to a very good standard there was not much cleaning that was required from our end we checked the dataset just in case for null values and did not find any abnormalities with the code.

EDA

We have performed 'Exploratory Data Analysis' before applying the machine learning models, by using graphical representations, to find abnormalities, and double-check assumptions along with exploring the data that is going to be used in the machine learning process, this also helps in selecting features. As depicted in the figure 2, out of 253680 entries, we discovered that 213703 patients have diabetes, 4631 people have prediabetes, and 35346 people do not have diabetes. It is quite clear to see that the dataset was significantly imbalanced.

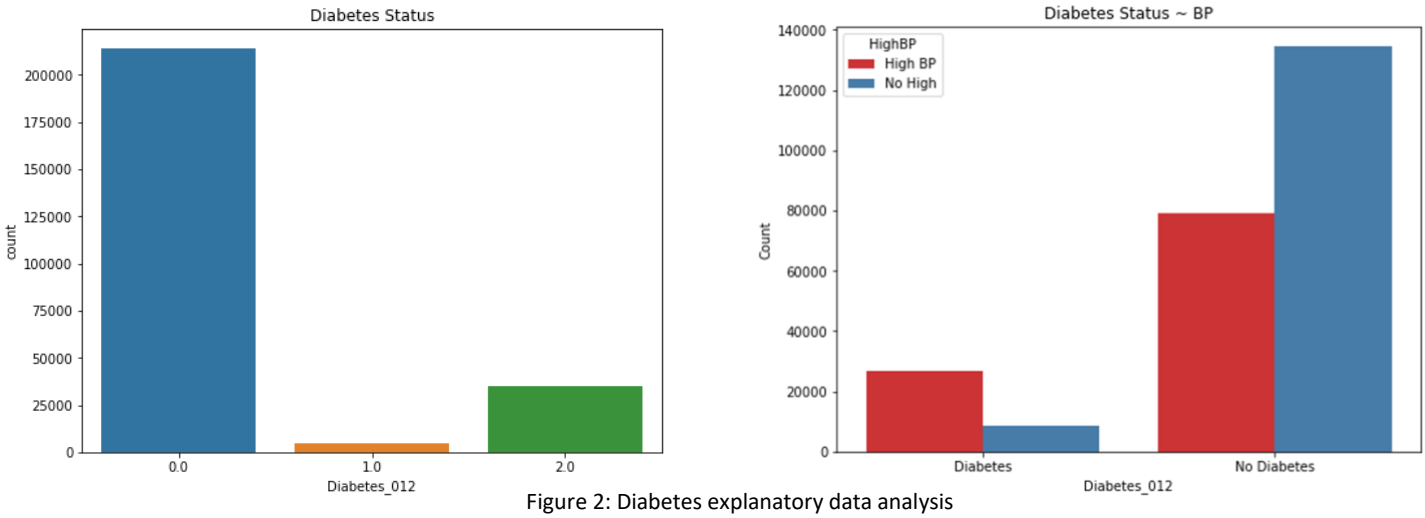


Figure 2: Diabetes explanatory data analysis

To fully comprehend the data, we performed Exploratory Data Analysis on specific columns. The above bar plot represents diabetes status with respect to blood pressure. We discovered that there are approximately 26604 people with diabetes who have high blood pressure and 8742 people with diabetes who do not have high blood pressure. There are 79312 people who do not have diabetes but have high blood pressure, and 134391 people who do not have diabetes but do not have high blood pressure.

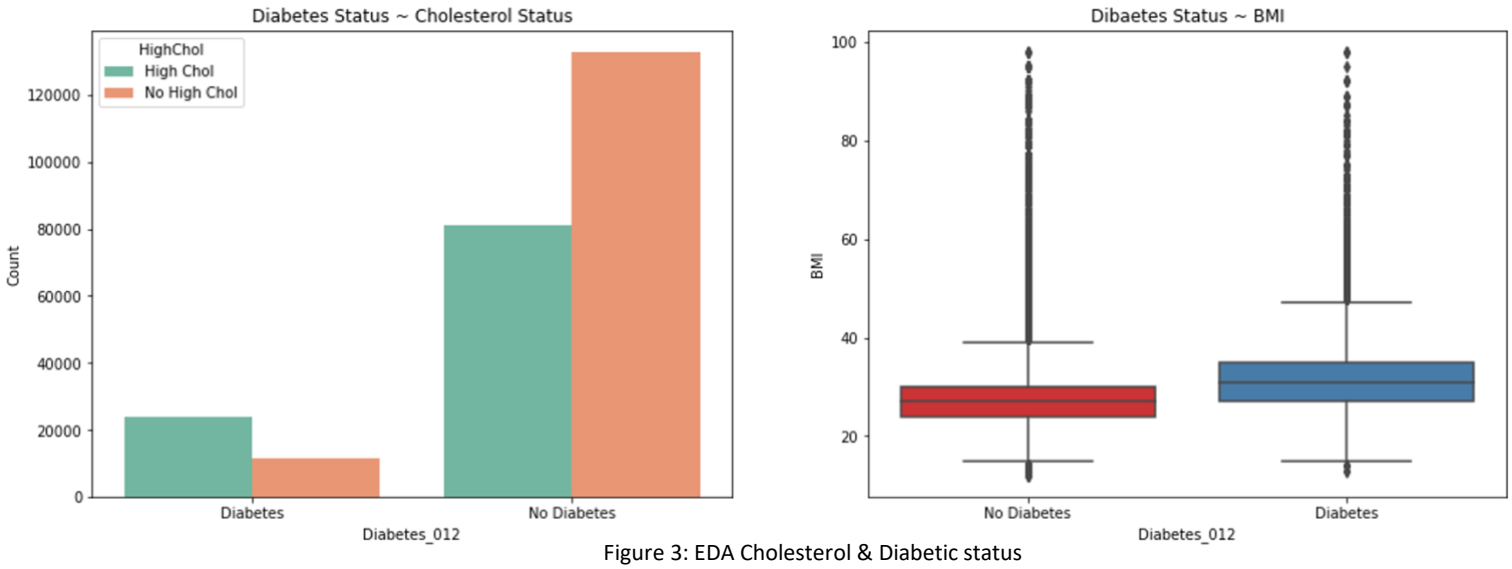


Figure 3: EDA Cholesterol & Diabetic status

We learned that there are around 23686 people with high cholesterol who also have diabetes, and 11660 people who do not have high cholesterol or diabetes. There are 81030 people with high cholesterol who do not have diabetes, and 132673 people who do not have diabetes but do have high cholesterol. The box plot in figure 3 represents diabetes status with respect to Body Mass Index. BMI was one of the few variables in our dataset that did not have a binary value. As a result, we developed a box plot to display and comprehend the data. We could see that the diabetic population's average BMI is between 25 and 32, with a maximum of 46 and a minimum of 12. The BMI value range for the nondiabetic population is between 15 and 40, with an average range of 22 to 27.

The bar chart in Appendix A represents diabetes status with respect to General Health. Where 'Excellent', 'good', 'fair', 'poor', and 'very good' are used to describe general health. There are 1140 people with diabetes and 'Excellent health.' There are 9790 people with diabetes and 'fair' health. Diabetes and 'good' health account for 13457 people, diabetes, and 'bad' health for 4578, and diabetes and 'very good' health for 6381.

43846 people who have no diabetes and are in 'excellent' health, 20755 people have no diabetes and are in 'fair' health, 6046 people have no diabetes and are in 'good' health, 7152 people have no diabetes and are in 'poor' health, and 81489 people have no diabetes and are in 'very good' health.

We also observed that people with diabetes who have no problem walking are 22225, while those with diabetes who have difficulties walking are 13121. There are 185434 people without diabetes who have no walking trouble, and there are 28269 people without diabetes who have walking difficulty. These results can be seen in Appendix A.

Data Cleaning

As data cleaning is such an important element of the process of increasing overall productivity, in the data set, we looked for NA/Null values. There were no Null values in the data set since we had a good data set. As a result, we didn't have to remove any Null values. The data set was cleaned to a significant level, with most columns containing binary values, with some exceptions like Body Mass Index, which cannot be binary, as we saw in the BMI section of the Exploratory Data Analysis. Following EDA, we calculated the correlation of all features with Diabetes 012 from which we chose the top 5 features with the highest correlation value from 23 columns. Blood Pressure, Cholesterol, BMI, General Health, and Walking Difficulty are the factors to consider as shown in the figure in Appendix A. Because we intended to keep our values binary, we also excluded Prediabetes from the dataset. 0.0 in the Diabetes 012 meant that the person did not have diabetes, 1.0 suggested that they were pre-diabetic, and 2.0

indicated that they had diabetes. We changed 2.0 to 1.0 after removing prediabetes, with 0.0 indicating no diabetes and 1.0 indicating diabetes as depicted in figure.

Technologies Used

For this project, we were able to get our hands on a variety of technologies to discover the most efficient yet effective solution. We ran on two, three, and four nodes using PySpark, which we installed locally on our machine. On a Jupyter notebook, we ran the code. For this assignment, we created three Machine Learning models. We used Spark MLib's Machine Learning libraries for this. We also used Amazon EMR and S3 to store the data along with running the models and comparing them to the models that were running locally to see how scalable our Machine Learning models were. We used the matplotlib and scikit-learn libraries for visualization.

Methodology

Before the team started the implementation of the machine learning algorithms it was crucial that the team and a plan and were checking the steps taken by team thus far have been accurate and precise. To help with this a flowchart has been created which can be seen in the figure below that maps the steps the team have carried out in this project. As depicted in the figure in Appendix B – in figure the data had been collected from Kaggle in a CSV format then the data was initialized and understood which has helped in the next process of cleaning the dataset. The data cleaning process has been explained above and is followed by the explanatory data analysis process which has also been explained supported by results. Next, we checked if the dataset was balanced if the dataset had not been balanced then we ran the

SMOTE function on the data that we had and balanced the dataset out. We then normalized the dataset and selected the key features for this project. After the feature selection the dataset was split into a ratio of 70:30 representing the training and testing set respectively. Three machine learning algorithms had been chosen and implemented using the PySpark Mlib library which included Logistic Regression, Random Forest, and Gradient Boosting. Finally, after fitting these models the results were evaluated and then further performance evaluation in terms of runtime and scalability of all three algorithms was conducted using amazon EMR.

Logistic Regression

There has already been a significant amount of research that has been conducted in the prediction of Diabetes specially in the discrimination between type 1 diabetes and type 2 diabetes which can be seen by the work of Lynam et al [4] . Logistic regression can be describes as one of the simplest and most popular machine learning algorithms to use when coming to solve binary classification problems. Logistic regression uses the sigmoid function to map out real world numerical values into a value between 0 and 1. In this project the team will be using Logistic Regression to predict/classify whether a patient has diabetes or not. Due to the popularity of using Logistic Regression in medical problems we chose this model for this project as several medical projects using machine learning have successfully implemented Logistic Regression and have found to derive good results. This technique is very useful when predicting the likelihood of event occurring and more importantly it can help in determining probabilities between two different classes. Some examples of Linear Regression are classification of emails being spam or not, prediction if it will rain or not and more medically related in the use of convolutional neural networks to identify if there is a tumor or not in a medical image.

Random Forest

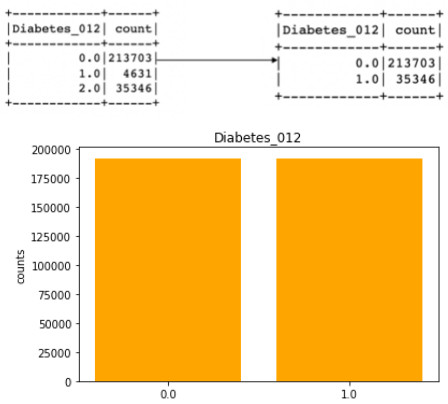
Random Forest can be described as an ensemble model as it consists of several decision trees within the classifier that each corresponds to a classification result. Using a majority voting system, the final classification result is based on the most popular classification decision result. This algorithm is very useful due to being an ensemble method it reduces the degree of overfitting due to the combination of multiple overfitting evaluators The Random Forest algorithm is another popular algorithm that has been used in several applications of machine learning in medicine. Wang et al [5] use the Random Forest classifier in their explanatory study of diabetes mellitus not only limited to diabetes Random Forest has also been used on other different medical problem sets such as the work of Alam et al who use the Random Forest model for medical data classification using feature selection [6].

Gradient Boosting Tree

The Gradient Boosting Tree algorithm is very similar to the AdaBoost algorithm as they both are ensemble approaches using decision trees however the main difference between the gradient boosting tree and the AdaBoost algorithms are that the gradient boosting tree has a depth larger than 1. The gradient boosting tree takes three crucial aspects into account the first being the optimization of the loss function, the second being a weak leaner making predictions and the most important aspect being the model adding weak learners to optimize the loss function and reduce the error.

Discussion

Before running the models, the team used the correlation function to select five features from the dataset. The team selected 'HighBP','HighChol','BMI','GenHlth','DiffWalk' and as the target variable the Diabetes column. A problem the team noticed was the balancing of the dataset from the exploratory data analysis the team had observed that dataset was highly imbalanced as 213703 data entries for people without diabetes and 35346 data entries for people with diabetes. Using the Synthetic Minority Oversampling technique, the team was able to balance the dataset with 192322 data entries for both classes – 'diabetes' and 'no diabetes'. This can be seen by the figure below which shows the implementation of the SMOTE function and the balancing of the dataset.




```
In [52]: from imblearn.over_sampling import SMOTE
         from imblearn.combine import SMOTEENN
         from collections import Counter

         sm = SMOTE(random_state=3)
         x_train_res, y_train_res = sm.fit_resample(X_train, Y_train)

In [59]: import pandas as pd
         dataframe_1 = pd.DataFrame(x_train_res,columns=['HighBP','HighChol','BMI','GenHlth','DiffWalk'])
         dataframe_2 = pd.DataFrame(y_train_res, columns = ['Diabetes_012'])
         result = dataframe_1.combine_first(dataframe_2)
         smote_1 = spark.createDataFrame(result)

In [60]: import matplotlib.pyplot as plt
         import numpy as np
         responses = smote_1.groupBy('Diabetes_012').count().collect()
         categories = [i[0] for i in responses]
         counts = [i[1] for i in responses]
         print(counts)
         ind = np.array(range(len(categories)))
         width = 0.35
         plt.bar (ind, counts, color='purple')
         plt.ylabel('counts' )
         plt.title('Diabetes_012')
         plt.xticks (ind, categories)
```

Figure 4: SMOTE to balance dataset

However, before the SMOTE the team wanted to experiment on the imbalanced dataset and calculate the accuracy. The Logistic Regression algorithm was implemented on the imbalanced dataset along with the Random Forest and Gradient Boosting Tree.

The team was surprised by the results as all three algorithms gave the similar results hence making it difficult to choose which algorithm performed the best but with an imbalanced

dataset, we found problems with the models the team had implemented. The results of these three algorithms are shown below for the imbalanced dataset with all the models not performing well on this dataset as poor precision and recall are shown for all three algorithms Random Forest showed the worse recall out of the three models when predicting diabetes with a recall score of 0.09. The same model showed the best recall score for predicting no diabetes. All three models produced and accuracy of 86% and hence why we used a representation of the confusion matrix to display our results.

After balancing the dataset, we got better results that represented all three models in a better angle on the diabetes health indicators dataset. The results can be seen in the figure below as out of all the three algorithms the team thought the Logistic Regression algorithm performed the best overall due to a good balance on the number of correctly predicted data entries. However, it is important to note that in the health care industry the false negative value is looked at more as medical practitioners would not want to falsely predict diabetes. Hence in this respect the Random Forest Classifier produced the least false negatives out of the three algorithms.

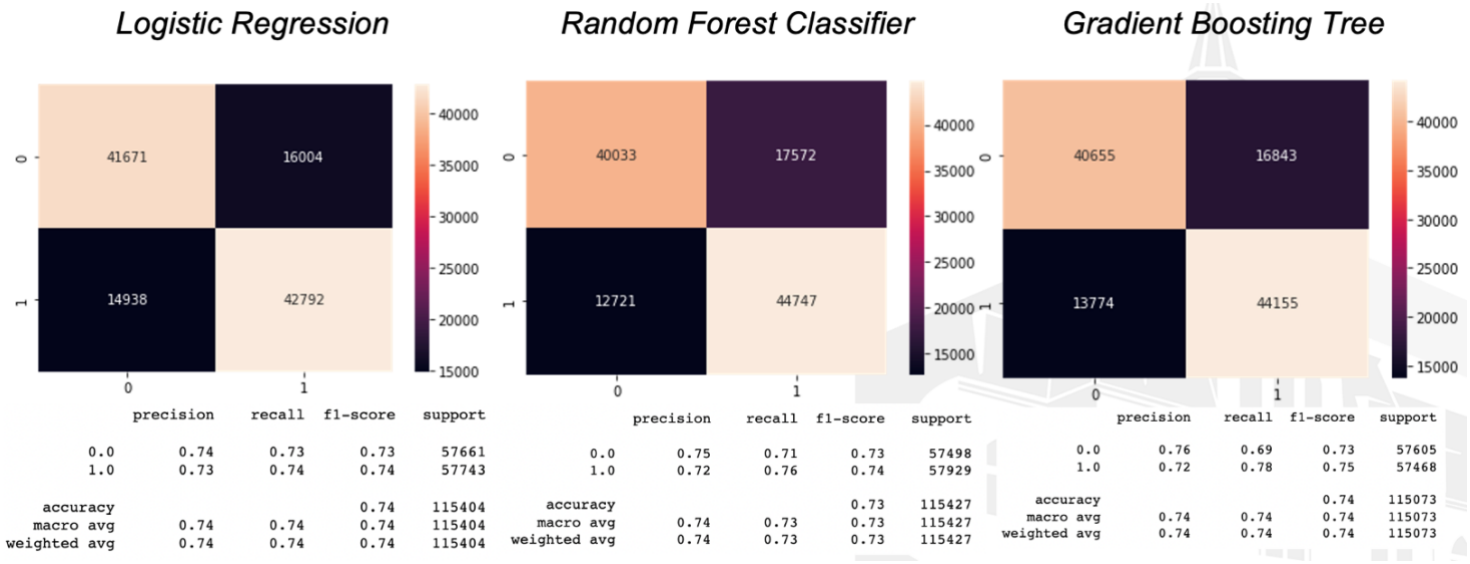
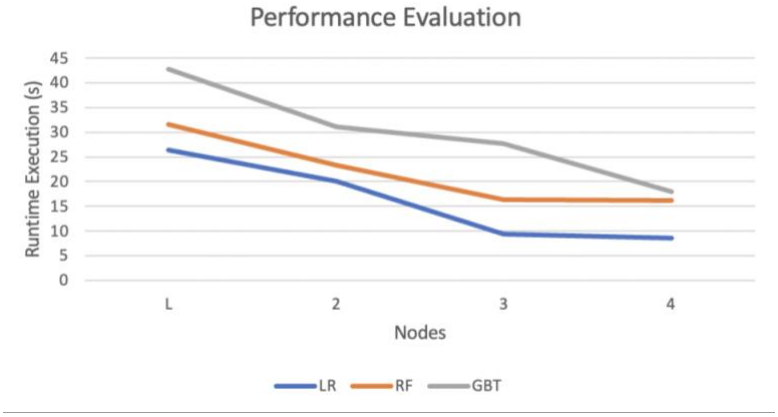


Figure 5: Balanced dataset results5

Performance Evaluation

To measure the performance of all these three algorithms on a balanced dataset we used AWS and specially Amazon EMR & S3 to run these models on clusters. Here we were able to measure the runtime of these three models and their scalability. Our first test was measuring the runtime against the number of nodes and whether the models are run on a local machine. From our results we found that the Logistic regression model ran the fastest out of the three algorithms on a local machine and on a cluster with 2, 3 and 4 nodes. This can be seen from the results in the figure below. As the number of nodes increase the runtime execution reduces; the Gradient Boosting Tree takes the longest time to run locally and on clusters with nodes ranging from 2 to 4. The Random Forest and Gradient Boosting Tree algorithms can be justified to perform slower than the Logistic Regression algorithm due to both of these algorithms being ensemble models, for example the Random Forest Classifier consists of several decisions tree which take their time to compute and predict an prediction and after that the majority voting is computed and similar reason is given for the Gradient Boosting Tree as the algorithm adds weak learners to make predications and optimize the loss function.



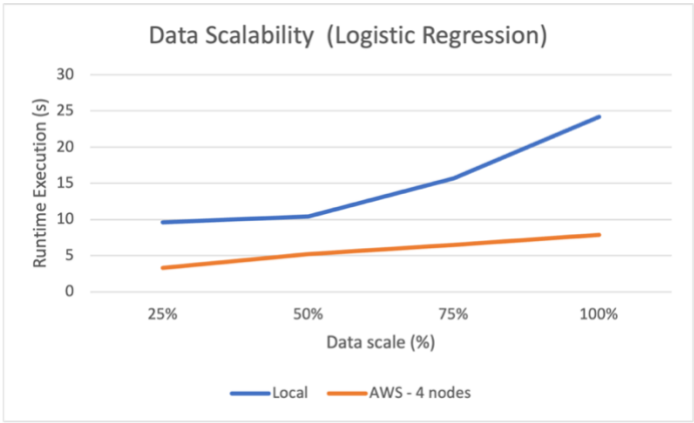


Figure 6: Performance Evaluation & Scalability

The next evaluation parameter is the runtime execution of the models against the scalability of the dataset. In this evaluation the models were ran on an AWS EMR cluster with four nodes the time taken for these models to run on 25%, 50%, 75% and 100% of the dataset were then noted and the curves can be seen on the graph in the figure below.

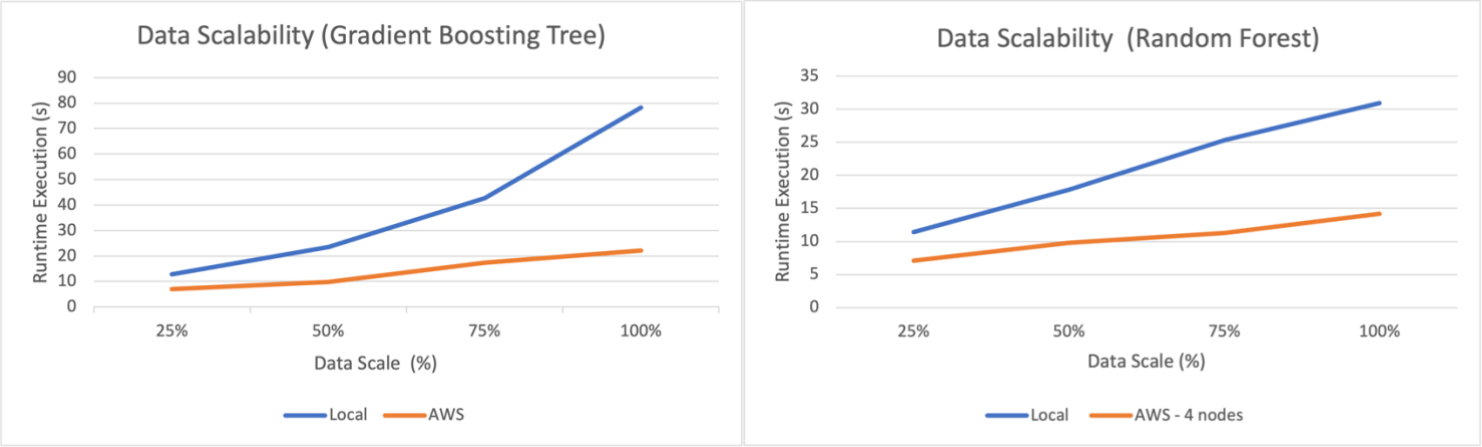


Figure 7: Data scalability using Gradient Boosting Tree and Random Forest

For all three models the results of the evaluation of the scalability showed that as the data scale increased the time to run these models took longer with intervals of a 25% in the increase of data. Out of these models all three took longer to run on a local machine compared to being ran on AWS. Hence this shows addition in value of running these three machine learning algorithms on AWS as the runtime decreases significantly for all three models and this can help in cutting costs of a business in the future if this model is deployed.

Conclusion

The data was successfully analysed and visualized, which can be seen in the Explanatory Data Analysis section. There is scope for future work to be conducted as the models can be evaluated on multiple platforms such as Microsoft Azure or Google Platform which may depict

a different runtime and scalability results. Moreover it is a possibility to run Principle Component Analysis and reduce the dimensionality of the dataset from 22 features, in our project we used the correlation feature and chose the top five correlated columns however in the future we could use PCA to see if the accuracy increases of the models. More importantly it would be extremely beneficial for us to add more data specifically with data entries that include people with diabetes as the current dataset is heavily imbalanced. Hyper-tuning the parameters is another technique to improve the accuracy of the results. Grid search is used to find the best parameters and accuracy. From our project we can conclude that we have learnt the following:

- Successfully able to use PySpark to analyse the Diabetes Health Indicators dataset.
- Implemented Logistic Regression, Random Forest Classifier and Gradient Boosting Tree gaining accuracy's of around 84%-86% on the three models on an imbalanced dataset however with poor recall and precision on the diabetes class.
- After implementing the SMOTE function we were able to gather a accuracy's of 74%-76% on all three of the algorithms with better recall and precision for the diabetes class.
- It can be concluded that it is better to run this PySpark script on AWS as the runtime execution is faster as the number of nodes increase. As the scale of the dataset increases the runtime execution of the model is faster than the script being ran locally.

References

[1] CDC, "National Diabetes Statistics Report," CDC, 2020.

[2] A. D. Association, "Economic Costs of Diabetes in the U.S. in 2017," Diabetes Care, 2018.

[3] S. Watson, "COVID-19 and Diabetes," WebMD, 2020.

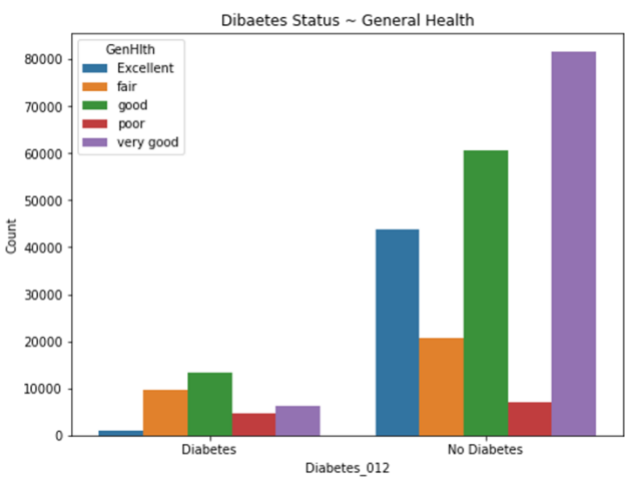
[4] A. L. Lynam, J. M. Dennis, K. R. Owen, R. A. Oram and B. M. S. 1. L. A. F. 1. Angus G Jones 1, "Logistic regression has similar performance to optimised machine learning algorithms in a clinical setting: application to the discrimination between type 1 and type 2 diabetes in young adults," Diagn Progn Res, 2020.

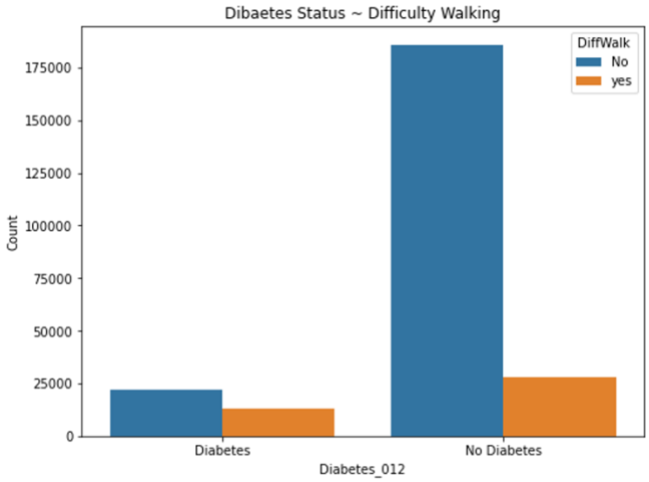
[5] X. Wang, M. Zhai, Z. Ren, H. Ren, M. Li, D. Quan, L. Chen and L. Qiu, "Exploratory study on classification of diabetes mellitus through a combined Random Forest Classifier".

[6] M. Z. Alam, M. S. Rahman and M. S. Rahman, "A Random Forest based predictor for medical data classification using feature ranking," Informatics in Medicine Unlocked, 2019.

Appendix

Appendix A – EDA Results



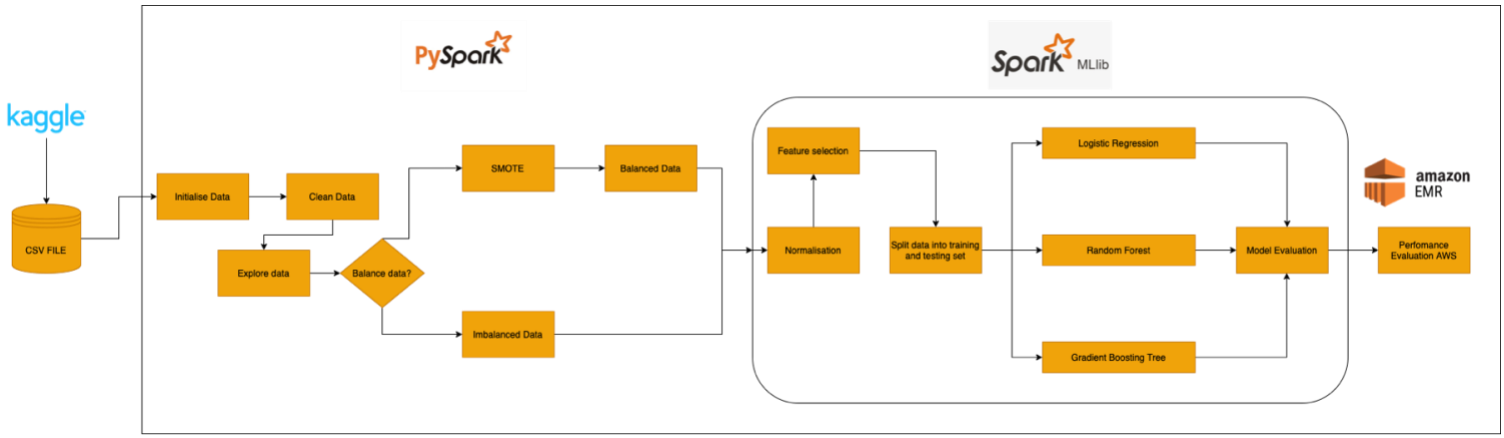


Appendix B – Correlation code

```
In [33]: for col in df.columns:
             print(col,":",df.stat.corr('Diabetes_012',col))
```

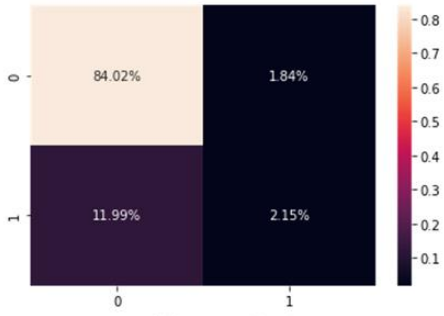
```
Diabetes_012 : 1.0
HighBP : 0.2715964243298824
HighChol : 0.20908491100576146
CholCheck : 0.0675464761160001
BMI : 0.22437947375839745
Smoker : 0.0629140950601668
Stroke : 0.10717866994339763
HeartDiseaseorAttack : 0.18027168633560287
PhysActivity : -0.12194716655035258
Fruits : -0.04219162985450684
Veggies : -0.05897159923470431
HvyAlcoholConsump : -0.05788191173818581
AnyHealthcare : 0.015410377239766794
NoDocbcCost : 0.035435685001551274
GenHlth : 0.3025866208859791
MentHlth : 0.07350676620655906
PhysHlth : 0.1762867357056404
DiffWalk : 0.22423912328804765
Sex : 0.031040163652105594
Age : 0.18502579410172637
Education : -0.1305169177115621
Income : -0.17148303762778555
```

Appendix C – Methodology Flowchart



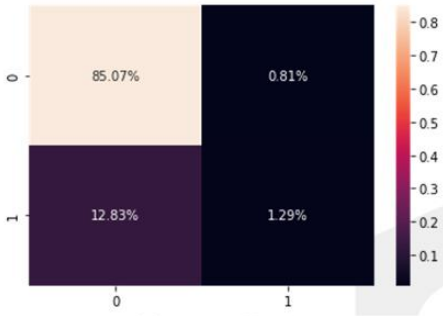
Appendix D – Imbalanced dataset results

Logistic Regression



	precision	recall	f1-score	support
0.0	0.88	0.98	0.92	63790
1.0	0.54	0.15	0.24	10509
accuracy			0.86	74299
macro avg	0.71	0.57	0.58	74299
weighted avg	0.83	0.86	0.83	74299

Random Forest Classifier



	precision	recall	f1-score	support
0.0	0.87	0.99	0.93	64344
1.0	0.61	0.09	0.16	10575
accuracy			0.86	74919
macro avg	0.74	0.54	0.54	74919
weighted avg	0.83	0.86	0.82	74919

Gradient Boosting Tree

	precision	recall	f1-score	support
0.0	0.87	0.98	0.93	64212
1.0	0.57	0.13	0.21	10537
accuracy			0.86	74749
macro avg	0.72	0.56	0.57	74749
weighted avg	0.83	0.86	0.82	74749

Appendix E – Code

In [1]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
import pandas as pd
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("spark").getOrCreate()
```

```
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
21/12/08 15:22:38 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

In [2]:
```
df = spark.read.csv('diabetes_012_health_indicators_BRFSS2015.csv', header = True, inferSchema = True)
```

Data Understanding

In this section of the code we will be attempting to understand the dataset that has been provided via Kaggle.

- https://www.kaggle.com/alexteboul/diabetes-health-indicators-dataset

In [3]:
```
df.show()
```

Diabetes_012	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	Veggies	HvyAlcoholConsump	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk	Sex	Age	Education	Income
0.0	1.0	1.0	1.0	40.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	5.0	18.0	15.0	1.0	0.0	9.0	4.0	3.0
0.0	0.0	0.0	0.0	25.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	3.0	0.0	0.0	0.0	0.0	7.0	6.0	1.0
0.0	1.0	1.0	1.0	28.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	5.0	30.0	30.0	1.0	0.0	9.0	4.0	8.0
0.0	1.0	0.0	1.0	27.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	2.0	0.0	0.0	0.0	0.0	11.0	3.0	6.0
0.0	1.0	1.0	1.0	24.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	2.0	3.0	0.0	0.0	0.0	11.0	5.0	4.0
0.0	1.0	1.0	1.0	25.0	1.0	0.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	2.0	0.0	2.0	0.0	1.0	10.0	6.0	8.0
0.0	1.0	0.0	1.0	30.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	3.0	0.0	14.0	0.0	0.0	9.0	6.0	7.0
0.0	1.0	1.0	1.0	25.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	3.0	0.0	0.0	1.0	0.0	11.0	4.0	4.0
2.0	1.0	1.0	1.0	30.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0	5.0	30.0	30.0	1.0	0.0	9.0	5.0	1.0
0.0	0.0	0.0	1.0	24.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	2.0	0.0	0.0	0.0	1.0	8.0	4.0	3.0
2.0	0.0	0.0	1.0	25.0	1.0	0.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	3.0	0.0	0.0	0.0	1.0	13.0	6.0	8.0
0.0	1.0	1.0	1.0	34.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	3.0	0.0	30.0	1.0	0.0	10.0	5.0	1.0
0.0	0.0	0.0	1.0	26.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	3.0	0.0	15.0	0.0	0.0	7.0	5.0	7.0
2.0	1.0	1.0	1.0	28.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	4.0	0.0	0.0	1.0	0.0	11.0	4.0	6.0
0.0	0.0	1.0	1.0	33.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	4.0	30.0	28.0	0.0	0.0	4.0	6.0	2.0
0.0	1.0	0.0	1.0	33.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	2.0	5.0	0.0	0.0	0.0	6.0	6.0	8.0
0.0	1.0	1.0	1.0	21.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	3.0	0.0	0.0	0.0	0.0	10.0	4.0	3.0
2.0	0.0	0.0	1.0	23.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	2.0	0.0	0.0	0.0	1.0	7.0	5.0	6.0
0.0	0.0	0.0	0.0	23.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	2.0	15.0	0.0	0.0	0.0	2.0	6.0	7.0
0.0	0.0	1.0	1.0	28.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	2.0	10.0	0.0	0.0	1.0	4.0	6.0	8.0

only showing top 20 rows

In [4]:
```
#Shape of the datadrame and counting the values in the Diabetes dataset.
print((df.count(),len(df.columns)))
df.groupby('Diabetes_012').count().show()
```

(253680, 22)

[Stage 6:> (0 + 6) / 6]

Diabetes_012	count
0.0	213703
1.0	4631
2.0	35346

In [5]:
```
#Removing prediabetes from the dataset and replacing the diabetes value from 2 to 1
from pyspark.sql import functions as F
df1 = df.filter(df.Diabetes_012!=1.0)
df1.show()
df2 = df1.na.replace([2.0], [1.0], 'Diabetes_012')
df2.groupby('Diabetes_012').count().show()
```

```
+-------------+------+--------+---------+----+------+------+--------------------+-------------+------+--------+--------------+----------+-----------+--------+-------+--------+--------+---+---+---------+------+
|Diabetes_012|HighBP|HighChol|CholCheck| BMI|Smoker|Stroke|HeartDiseaseorAttack|PhysActivity|Fruits|Veggies|HvyAlcoholConsump|AnyHealthcare|NoDocbcCost|GenHlth|MentHlth|PhysHlth|DiffWalk|Sex|Age|Education|Income|
+-------------+------+--------+---------+----+------+------+--------------------+-------------+------+--------+--------------+----------+-----------+--------+-------+--------+--------+---+---+---------+------+
|          0.0|   1.0|     1.0|      1.0|40.0|   1.0|   0.0|                 0.0|          0.0|   0.0|     1.0|           0.0|       1.0|        0.0|     5.0|   18.0|    15.0|     1.0|0.0|9.0|      4.0|   3.0|
|          0.0|   0.0|     0.0|      0.0|25.0|   1.0|   0.0|                 0.0|          1.0|   0.0|     0.0|           0.0|       0.0|        1.0|     3.0|    0.0|     0.0|     0.0|0.0|7.0|      6.0|   1.0|
|          0.0|   1.0|     1.0|      1.0|28.0|   0.0|   0.0|                 0.0|          0.0|   1.0|     0.0|           0.0|       1.0|        1.0|     5.0|   30.0|    30.0|     1.0|0.0|9.0|      4.0|   8.0|
|          0.0|   1.0|     0.0|      1.0|27.0|   0.0|   0.0|                 0.0|          1.0|   1.0|     1.0|           0.0|       1.0|        0.0|     2.0|    0.0|     0.0|     0.0|0.0|11.0|      3.0|   6.0|
|          0.0|   1.0|     1.0|      1.0|24.0|   0.0|   0.0|                 0.0|          1.0|   1.0|     1.0|           0.0|       1.0|        0.0|     2.0|    3.0|     0.0|     0.0|0.0|11.0|      5.0|   4.0|
|          0.0|   1.0|     1.0|      1.0|25.0|   1.0|   0.0|                 0.0|          1.0|   1.0|     1.0|           0.0|       1.0|        0.0|     2.0|    0.0|     2.0|     0.0|1.0|10.0|      6.0|   8.0|
|          0.0|   1.0|     0.0|      1.0|30.0|   1.0|   0.0|                 0.0|          0.0|   0.0|     0.0|           0.0|       1.0|        0.0|     3.0|    0.0|    14.0|     0.0|0.0|9.0|      6.0|   7.0|
|          0.0|   1.0|     1.0|      1.0|25.0|   1.0|   0.0|                 0.0|          1.0|   0.0|     1.0|           0.0|       1.0|        0.0|     3.0|    0.0|     0.0|     1.0|0.0|11.0|      4.0|   4.0|
|          2.0|   1.0|     1.0|      1.0|30.0|   1.0|   0.0|                 1.0|          0.0|   1.0|     1.0|           0.0|       1.0|        0.0|     5.0|   30.0|    30.0|     1.0|0.0|9.0|      5.0|   1.0|
|          0.0|   0.0|     0.0|      1.0|24.0|   0.0|   0.0|                 0.0|          0.0|   0.0|     1.0|           0.0|       1.0|        0.0|     2.0|    0.0|     0.0|     0.0|1.0|8.0|      4.0|   3.0|
|          2.0|   0.0|     0.0|      1.0|25.0|   1.0|   0.0|                 0.0|          1.0|   1.0|     1.0|           0.0|       1.0|        0.0|     3.0|    0.0|     0.0|     0.0|1.0|13.0|      6.0|   8.0|
|          0.0|   1.0|     1.0|      1.0|34.0|   1.0|   0.0|                 0.0|          0.0|   1.0|     1.0|           0.0|       1.0|        0.0|     3.0|    0.0|     0.0|     0.0|1.0|10.0|      5.0|   1.0|
|          0.0|   0.0|     0.0|      1.0|26.0|   1.0|   0.0|                 0.0|          0.0|   0.0|     1.0|           0.0|       1.0|        0.0|     3.0|    0.0|    15.0|     0.0|0.0|7.0|      5.0|   7.0|
|          2.0|   1.0|     1.0|      1.0|28.0|   0.0|   0.0|                 0.0|          0.0|   0.0|     1.0|           0.0|       1.0|        0.0|     4.0|    0.0|     0.0|     1.0|0.0|11.0|      4.0|   6.0|
|          0.0|   0.0|     1.0|      1.0|33.0|   1.0|   1.0|                 0.0|          1.0|   0.0|     1.0|           0.0|       1.0|        1.0|     4.0|   30.0|    28.0|     0.0|0.0|4.0|      6.0|   2.0|
|          0.0|   1.0|     0.0|      1.0|33.0|   0.0|   0.0|                 0.0|          1.0|   0.0|     0.0|           0.0|       1.0|        0.0|     2.0|    5.0|     0.0|     0.0|0.0|6.0|      6.0|   8.0|
|          0.0|   1.0|     1.0|      1.0|21.0|   0.0|   0.0|                 0.0|          1.0|   1.0|     1.0|           0.0|       1.0|        0.0|     3.0|    0.0|     0.0|     0.0|0.0|10.0|      4.0|   3.0|
|          2.0|   0.0|     0.0|      1.0|23.0|   1.0|   0.0|                 0.0|          1.0|   0.0|     0.0|           0.0|       1.0|        0.0|     2.0|    0.0|     0.0|     0.0|1.0|7.0|      5.0|   6.0|
|          0.0|   0.0|     0.0|      0.0|23.0|   0.0|   0.0|                 0.0|          0.0|   0.0|     1.0|           0.0|       1.0|        0.0|     2.0|   15.0|     0.0|     0.0|0.0|2.0|      6.0|   7.0|
|          0.0|   0.0|     1.0|      1.0|28.0|   0.0|   0.0|                 0.0|          0.0|   0.0|     0.0|           1.0|       1.0|        0.0|     2.0|   10.0|     0.0|     0.0|1.0|4.0|      6.0|   8.0|
+-------------+------+--------+---------+----+------+------+--------------------+-------------+------+--------+--------------+----------+-----------+--------+-------+--------+--------+---+---+---------+------+
only showing top 20 rows

+-------------+------+
|Diabetes_012| count|
+-------------+------+
|          0.0|213703|
|          1.0| 35346|
+-------------+------+
```

In [9]:
```
for col in df.columns:
    print(col,"\t",df.stat.corr('Diabetes_012',col))
```

```
Diabetes_012 : 1.0
HighBP : 0.2715964243298824
HighChol : 0.2090849110576146
CholCheck : 0.06754647411600013
BMI : 0.2243794737583975
Smoker : 0.0629140950601668
Stroke : 0.1071786699439764
HeartDiseaseorAttack : 0.1802716863560287
PhysActivity : -0.1219471665035258
Fruits : -0.0421916298545068
Veggies : -0.0589715992347043
HvyAlcoholConsump : -0.0578819117381858
AnyHealthcare : 0.0154103772397667
NoDocbcCost : 0.0354356850015512
GenHlth : 0.3025866208859791
MentHlth : 0.0735067662065906
PhysHlth : 0.1762867357056404
DiffWalk : 0.2242391232880476
Sex : 0.0310401636522105
Age : 0.1850257941017263
Education : -0.1305169177115621
Income : -0.1714830762778558
```

In [94]:
```
df.printSchema()
```

```
root
 |-- Diabetes_012: double (nullable = true)
 |-- HighBP: double (nullable = true)
 |-- HighChol: double (nullable = true)
 |-- CholCheck: double (nullable = true)
 |-- BMI: double (nullable = true)
 |-- Smoker: double (nullable = true)
 |-- Stroke: double (nullable = true)
 |-- HeartDiseaseorAttack: double (nullable = true)
 |-- PhysActivity: double (nullable = true)
 |-- Fruits: double (nullable = true)
 |-- Veggies: double (nullable = true)
 |-- HvyAlcoholConsump: double (nullable = true)
 |-- AnyHealthcare: double (nullable = true)
 |-- NoDocbcCost: double (nullable = true)
 |-- GenHlth: double (nullable = true)
 |-- MentHlth: double (nullable = true)
 |-- PhysHlth: double (nullable = true)
 |-- DiffWalk: double (nullable = true)
 |-- Sex: double (nullable = true)
 |-- Age: double (nullable = true)
 |-- Education: double (nullable = true)
 |-- Income: double (nullable = true)
```

In [95]:
```
#Data cleaning
def dataCleaning(df,print_result):
    for col in df.columns:
        if print_result==True:
            print(col,":",df[df[col].isNull()].count())
```

In [96]:
```
print("Finding NULL values")
dataCleaning(df,True)
```

```
Finding NULL values
Diabetes_012 : 0
HighBP : 0
HighChol : 0
CholCheck : 0
BMI : 0
Smoker : 0
Stroke : 0
HeartDiseaseorAttack : 0
PhysActivity : 0
Fruits : 0
Veggies : 0
HvyAlcoholConsump : 0
AnyHealthcare : 0
NoDocbcCost : 0
GenHlth : 0
MentHlth : 0
PhysHlth : 0
DiffWalk : 0
Sex : 0
Age : 0
Education : 0
Income : 0
```

```
#Removing prediabetes from the dataset and replacing the diabetes value from 2 to 1
from pyspark.sql import functions as F
df1 = df.filter(df.Diabetes_012!=1.0)
df1.show()
df2 = df1.na.replace([2.0], [1.0], 'Diabetes_012')
df2.groupby('Diabetes_012').count().show()
```

```
+------------+------+--------+---------+----+------+------+--------------------+------------+------+-------+-----------------+-------------+-----------+-------+--------+--------+--------+---+----+---------+------+
|Diabetes_012|HighBP|HighChol|CholCheck| BMI|Smoker|Stroke|HeartDiseaseorAttack|PhysActivity|Fruits|Veggies|HvyAlcoholConsump|AnyHealthcare|NoDocbcCost|GenHlth|MentHlth|PhysHlth|DiffWalk|Sex| Age|Education|Income|
+------------+------+--------+---------+----+------+------+--------------------+------------+------+-------+-----------------+-------------+-----------+-------+--------+--------+--------+---+----+---------+------+
|         0.0|   1.0|     1.0|      1.0|40.0|   1.0|   0.0|                 0.0|         0.0|   0.0|    1.0|              0.0|          1.0|        0.0|    5.0|    18.0|    15.0|     1.0|0.0| 9.0|      4.0|   3.0|
|         0.0|   0.0|     0.0|      0.0|25.0|   1.0|   0.0|                 0.0|         1.0|   0.0|    0.0|              0.0|          0.0|        1.0|    3.0|     0.0|     0.0|     0.0|0.0| 7.0|      6.0|   1.0|
|         0.0|   1.0|     1.0|      1.0|28.0|   0.0|   0.0|                 0.0|         0.0|   1.0|    0.0|              0.0|          1.0|        1.0|    5.0|    30.0|    30.0|     1.0|0.0| 9.0|      4.0|   8.0|
|         0.0|   1.0|     0.0|      1.0|27.0|   0.0|   0.0|                 0.0|         1.0|   1.0|    1.0|              0.0|          1.0|        0.0|    2.0|     0.0|     0.0|     0.0|0.0|11.0|      3.0|   6.0|
|         0.0|   1.0|     1.0|      1.0|24.0|   0.0|   0.0|                 0.0|         1.0|   1.0|    1.0|              0.0|          1.0|        0.0|    2.0|     3.0|     0.0|     0.0|0.0|11.0|      5.0|   4.0|
|         0.0|   1.0|     1.0|      1.0|25.0|   1.0|   0.0|                 0.0|         1.0|   1.0|    1.0|              0.0|          1.0|        0.0|    2.0|     0.0|     2.0|     0.0|1.0|10.0|      6.0|   8.0|
|         0.0|   1.0|     0.0|      1.0|30.0|   1.0|   0.0|                 0.0|         0.0|   0.0|    0.0|              0.0|          1.0|        0.0|    3.0|     0.0|    14.0|     0.0|0.0| 9.0|      6.0|   7.0|
|         0.0|   1.0|     1.0|      1.0|25.0|   1.0|   0.0|                 0.0|         1.0|   0.0|    1.0|              0.0|          1.0|        0.0|    3.0|     0.0|     0.0|     1.0|0.0|11.0|      4.0|   4.0|
|         2.0|   1.0|     1.0|      1.0|30.0|   1.0|   0.0|                 1.0|         0.0|   1.0|    1.0|              0.0|          1.0|        0.0|    5.0|    30.0|    30.0|     1.0|0.0| 9.0|      5.0|   1.0|
|         0.0|   0.0|     0.0|      1.0|24.0|   0.0|   0.0|                 0.0|         0.0|   0.0|    1.0|              0.0|          1.0|        0.0|    2.0|     0.0|     0.0|     0.0|1.0| 8.0|      4.0|   3.0|
|         2.0|   0.0|     0.0|      1.0|25.0|   1.0|   0.0|                 0.0|         1.0|   1.0|    1.0|              0.0|          1.0|        0.0|    3.0|     0.0|     0.0|     0.0|1.0|13.0|      6.0|   8.0|
|         0.0|   1.0|     1.0|      1.0|34.0|   1.0|   0.0|                 0.0|         0.0|   1.0|    1.0|              0.0|          1.0|        0.0|    3.0|     0.0|    30.0|     1.0|0.0|10.0|      5.0|   1.0|
|         0.0|   0.0|     0.0|      1.0|26.0|   1.0|   0.0|                 0.0|         0.0|   0.0|    1.0|              0.0|          1.0|        0.0|    3.0|     0.0|    15.0|     0.0|0.0| 7.0|      5.0|   7.0|
|         2.0|   1.0|     1.0|      1.0|28.0|   0.0|   0.0|                 0.0|         0.0|   0.0|    1.0|              0.0|          1.0|        0.0|    4.0|     0.0|     0.0|     1.0|0.0|11.0|      4.0|   6.0|
|         0.0|   0.0|     1.0|      1.0|33.0|   1.0|   1.0|                 0.0|         1.0|   0.0|    1.0|              0.0|          1.0|        1.0|    4.0|    30.0|    28.0|     0.0|0.0| 4.0|      6.0|   2.0|
|         0.0|   1.0|     0.0|      1.0|33.0|   0.0|   0.0|                 0.0|         1.0|   0.0|    0.0|              0.0|          1.0|        0.0|    2.0|     5.0|     0.0|     0.0|0.0| 6.0|      6.0|   8.0|
|         0.0|   1.0|     1.0|      1.0|21.0|   0.0|   0.0|                 0.0|         1.0|   1.0|    1.0|              0.0|          1.0|        0.0|    3.0|     0.0|     0.0|     0.0|0.0|10.0|      4.0|   3.0|
|         2.0|   0.0|     0.0|      1.0|23.0|   1.0|   0.0|                 0.0|         1.0|   0.0|    0.0|              0.0|          1.0|        0.0|    2.0|     0.0|     0.0|     0.0|1.0| 7.0|      5.0|   6.0|
|         0.0|   0.0|     0.0|      0.0|23.0|   0.0|   0.0|                 0.0|         0.0|   0.0|    1.0|              0.0|          1.0|        0.0|    2.0|    15.0|     0.0|     0.0|0.0| 2.0|      6.0|   7.0|
|         0.0|   0.0|     1.0|      1.0|28.0|   0.0|   0.0|                 0.0|         0.0|   0.0|    0.0|              1.0|          1.0|        0.0|    2.0|    10.0|     0.0|     0.0|1.0| 4.0|      6.0|   8.0|
+------------+------+--------+---------+----+------+------+--------------------+------------+------+-------+-----------------+-------------+-----------+-------+--------+--------+--------+---+----+---------+------+
only showing top 20 rows

+------------+------+
|Diabetes_012| count|
+------------+------+
|         0.0|213703|
|         1.0| 35346|
+------------+------+
```

```
for col in df.columns:
    print(col,":",df.stat.corr('Diabetes_012',col))
```

```
Diabetes_012 : 1.0
HighBP : 0.2715964243298824
HighChol : 0.20908491100576146
CholCheck : 0.06754647611600013
BMI : 0.2243794737583975
Smoker : 0.0629140950601668
Stroke : 0.10717866994339764
HeartDiseaseorAttack : 0.18027168633560287
PhysActivity : -0.12194716655035258
Fruits : -0.04219162985450684
Veggies : -0.058971599234704304
HvyAlcoholConsump : -0.05788191173818581
AnyHealthcare : 0.015410377239766794
NoDocbcCost : 0.03543568500155128
GenHlth : 0.3025866208859791
MentHlth : 0.07350676620655906
PhysHlth : 0.17628673570564043
DiffWalk : 0.22423912328804765
Sex : 0.031040163652105594
Age : 0.18502579410172637
Education : -0.1305169177115621
Income : -0.17148303762778558
```

```
df2.show()
```

```
+------------+------+--------+---------+----+------+------+--------------------+------------+------+-------+-----------------+-------------+-----------+-------+--------+--------+--------+---+----+---------+------+
|Diabetes_012|HighBP|HighChol|CholCheck| BMI|Smoker|Stroke|HeartDiseaseorAttack|PhysActivity|Fruits|Veggies|HvyAlcoholConsump|AnyHealthcare|NoDocbcCost|GenHlth|MentHlth|PhysHlth|DiffWalk|Sex| Age|Education|Income|
+------------+------+--------+---------+----+------+------+--------------------+------------+------+-------+-----------------+-------------+-----------+-------+--------+--------+--------+---+----+---------+------+
|         0.0|   1.0|     1.0|      1.0|40.0|   1.0|   0.0|                 0.0|         0.0|   0.0|    1.0|              0.0|          1.0|        0.0|    5.0|    18.0|    15.0|     1.0|0.0| 9.0|      4.0|   3.0|
|         0.0|   0.0|     0.0|      0.0|25.0|   1.0|   0.0|                 0.0|         1.0|   0.0|    0.0|              0.0|          0.0|        1.0|    3.0|     0.0|     0.0|     0.0|0.0| 7.0|      6.0|   1.0|
|         0.0|   1.0|     1.0|      1.0|28.0|   0.0|   0.0|                 0.0|         0.0|   1.0|    0.0|              0.0|          1.0|        1.0|    5.0|    30.0|    30.0|     1.0|0.0| 9.0|      4.0|   8.0|
|         0.0|   1.0|     0.0|      1.0|27.0|   0.0|   0.0|                 0.0|         1.0|   1.0|    1.0|              0.0|          1.0|        0.0|    2.0|     0.0|     0.0|     0.0|0.0|11.0|      3.0|   6.0|
|         0.0|   1.0|     1.0|      1.0|24.0|   0.0|   0.0|                 0.0|         1.0|   1.0|    1.0|              0.0|          1.0|        0.0|    2.0|     3.0|     0.0|     0.0|0.0|11.0|      5.0|   4.0|
|         0.0|   1.0|     1.0|      1.0|25.0|   1.0|   0.0|                 0.0|         1.0|   1.0|    1.0|              0.0|          1.0|        0.0|    2.0|     0.0|     2.0|     0.0|1.0|10.0|      6.0|   8.0|
|         0.0|   1.0|     0.0|      1.0|30.0|   1.0|   0.0|                 0.0|         0.0|   0.0|    0.0|              0.0|          1.0|        0.0|    3.0|     0.0|    14.0|     0.0|0.0| 9.0|      6.0|   7.0|
|         0.0|   1.0|     1.0|      1.0|25.0|   1.0|   0.0|                 0.0|         1.0|   0.0|    1.0|              0.0|          1.0|        0.0|    3.0|     0.0|     0.0|     1.0|0.0|11.0|      4.0|   4.0|
|         1.0|   1.0|     1.0|      1.0|30.0|   1.0|   0.0|                 1.0|         0.0|   1.0|    1.0|              0.0|          1.0|        0.0|    5.0|    30.0|    30.0|     1.0|0.0| 9.0|      5.0|   1.0|
|         0.0|   0.0|     0.0|      1.0|24.0|   0.0|   0.0|                 0.0|         0.0|   0.0|    1.0|              0.0|          1.0|        0.0|    2.0|     0.0|     0.0|     0.0|1.0| 8.0|      4.0|   3.0|
|         1.0|   0.0|     0.0|      1.0|25.0|   1.0|   0.0|                 0.0|         1.0|   1.0|    1.0|              0.0|          1.0|        0.0|    3.0|     0.0|     0.0|     0.0|1.0|13.0|      6.0|   8.0|
|         0.0|   1.0|     1.0|      1.0|34.0|   1.0|   0.0|                 0.0|         0.0|   1.0|    1.0|              0.0|          1.0|        0.0|    3.0|     0.0|    30.0|     1.0|0.0|10.0|      5.0|   1.0|
|         0.0|   0.0|     0.0|      1.0|26.0|   1.0|   0.0|                 0.0|         0.0|   0.0|    1.0|              0.0|          1.0|        0.0|    3.0|     0.0|    15.0|     0.0|0.0| 7.0|      5.0|   7.0|
|         1.0|   1.0|     1.0|      1.0|28.0|   0.0|   0.0|                 0.0|         0.0|   0.0|    1.0|              0.0|          1.0|        0.0|    4.0|     0.0|     0.0|     1.0|0.0|11.0|      4.0|   6.0|
|         0.0|   0.0|     1.0|      1.0|33.0|   1.0|   1.0|                 0.0|         1.0|   0.0|    1.0|              0.0|          1.0|        1.0|    4.0|    30.0|    28.0|     0.0|0.0| 4.0|      6.0|   2.0|
|         0.0|   1.0|     0.0|      1.0|33.0|   0.0|   0.0|                 0.0|         1.0|   0.0|    0.0|              0.0|          1.0|        0.0|    2.0|     5.0|     0.0|     0.0|0.0| 6.0|      6.0|   8.0|
|         0.0|   1.0|     1.0|      1.0|21.0|   0.0|   0.0|                 0.0|         1.0|   1.0|    1.0|              0.0|          1.0|        0.0|    3.0|     0.0|     0.0|     0.0|0.0|10.0|      4.0|   3.0|
|         1.0|   0.0|     0.0|      1.0|23.0|   1.0|   0.0|                 0.0|         1.0|   0.0|    0.0|              0.0|          1.0|        0.0|    2.0|     0.0|     0.0|     0.0|1.0| 7.0|      5.0|   6.0|
|         0.0|   0.0|     0.0|      0.0|23.0|   0.0|   0.0|                 0.0|         0.0|   0.0|    1.0|              0.0|          1.0|        0.0|    2.0|    15.0|     0.0|     0.0|0.0| 2.0|      6.0|   7.0|
|         0.0|   0.0|     1.0|      1.0|28.0|   0.0|   0.0|                 0.0|         0.0|   0.0|    0.0|              1.0|          1.0|        0.0|    2.0|    10.0|     0.0|     0.0|1.0| 4.0|      6.0|   8.0|
+------------+------+--------+---------+----+------+------+--------------------+------------+------+-------+-----------------+-------------+-----------+-------+--------+--------+--------+---+----+---------+------+
only showing top 20 rows
```

Explanatory Data Analysis

```
eda1 = df2.toPandas()
```

```
eda1.groupby('Diabetes_012').count()
```

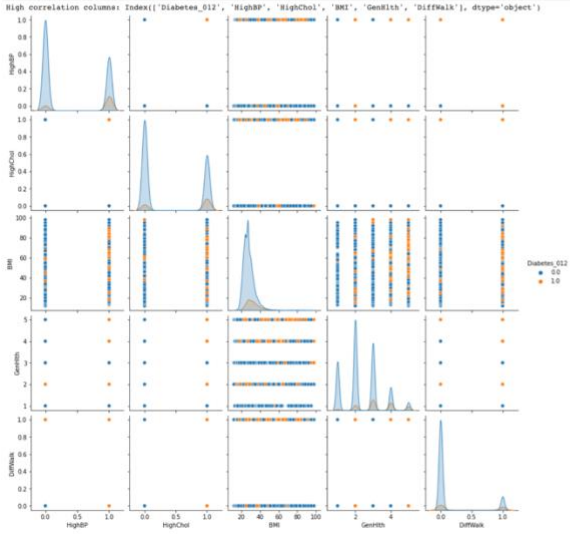
	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	Veggies	...	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk	Sex	Age	Education	Income
Diabetes_012																					
0.0	213703	213703	213703	213703	213703	213703	213703	213703	213703	213703	...	213703	213703	213703	213703	213703	213703	213703	213703	213703	213703
1.0	35346	35346	35346	35346	35346	35346	35346	35346	35346	35346	...	35346	35346	35346	35346	35346	35346	35346	35346	35346	35346

2 rows × 21 columns

```
#visualize diabetes status
plt.figure(figsize = (8,6))
sns.countplot(eda1['Diabetes_012'], palette="rocket")
plt.title("Diabetes Status")
plt.show()
```

In [22]:
```
eda1.corr()['Diabetes_012']
max_corr = eda1.corr()['Diabetes_012'][eda1.corr()['Diabetes_012'] > 0.2].keys()

max_corr = eda1.corr()['Diabetes_012'][eda1.corr()['Diabetes_012'] > 0.2].keys()

print("High correlation columns:", max_corr)
max_corr_cols = max_corr[1:]

sns.pairplot(eda1, x_vars = max_corr_cols, y_vars = max_corr_cols, hue='Diabetes_012', height=2.5, palette="rocket")
plt.show()
```

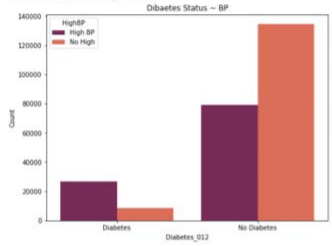
```
High correlation columns: Index(['Diabetes_012', 'HighBP', 'HighChol', 'BMI', 'GenHlth', 'DiffWalk'], dtype='object')
```



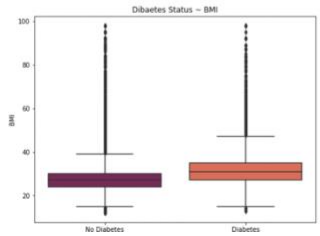
In [45]:
```
eda2 = eda1
eda2.Diabetes_012[eda2['Diabetes_012'] == 0.0] = 'No Diabetes'
eda2.Diabetes_012[eda2['Diabetes_012'] == 1.0] = 'Diabetes'


eda2.HighBP[eda2['HighBP'] == 0.0] = 'No High'
eda2.HighBP[eda2['HighBP'] == 1.0] = 'High BP'

eda2.HighChol[eda2['HighChol'] == 0.0] = 'No High Chol'
eda2.HighChol[eda2['HighChol'] == 1.0] = 'High Chol'

eda2.GenHlth[eda2['GenHlth'] == 1.0] = 'Excellent'
eda2.GenHlth[eda2['GenHlth'] == 2.0] = 'very good'
eda2.GenHlth[eda2['GenHlth'] == 3.0] = 'good'
eda2.GenHlth[eda2['GenHlth'] == 4.0] = 'fair'
eda2.GenHlth[eda2['GenHlth'] == 5.0] = 'poor'

eda2.DiffWalk[eda2['DiffWalk'] == 0.0] = 'No'
eda2.DiffWalk[eda2['DiffWalk'] == 1.0] = 'yes'
```

In [47]:
```
#group diabetes status & BP
dia_bp = eda2.groupby(['Diabetes_012', 'HighBP']).size().reset_index(name = 'Count')
print(dia_bp)

#visualize diabetes status ~ BP
plt.figure(figsize = (8,6))
sns.barplot(x = 'Diabetes_012', y = 'Count', hue = 'HighBP', data = dia_bp, palette="rocket")
plt.title("Dibaetes Status ~ BP")
plt.show()
```

```
  Diabetes_012   HighBP   Count
0     Diabetes  High BP   26604
1     Diabetes  No High    8742
2  No Diabetes  High BP   79312
3  No Diabetes  No High  134391
```



In [48]:
```
#visualize diabetes status ~ BMI
plt.figure(figsize = (8,6))
sns.boxplot(data = eda2, x = 'Diabetes_012', y = 'BMI', palette="rocket")
plt.title("Dibaetes Status ~ BMI")
plt.show()
```



In [46]:
```
eda2 = eda2[['Diabetes_012', 'HighBP', 'HighChol', 'BMI', 'GenHlth', 'DiffWalk']]
eda2
```

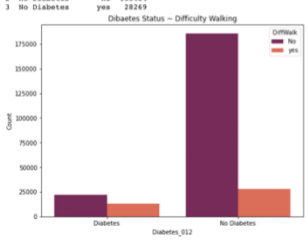
Out[46]:

	Diabetes_012	HighBP	HighChol	BMI	GenHlth	DiffWalk
0	No Diabetes	High BP	High Chol	40.0	poor	yes
1	No Diabetes	No High	No High Chol	25.0	good	No
2	No Diabetes	High BP	High Chol	28.0	poor	yes
3	No Diabetes	High BP	No High Chol	27.0	very good	No
4	No Diabetes	High BP	High Chol	24.0	very good	No
...	...	...	...	...	...	...
249044	No Diabetes	High BP	High Chol	45.0	good	No
249045	Diabetes	High BP	High Chol	18.0	fair	yes
249046	No Diabetes	No High	No High Chol	28.0	Excellent	No
249047	No Diabetes	High BP	No High Chol	23.0	good	No
249048	Diabetes	High BP	High Chol	25.0	very good	No

249049 rows × 6 columns

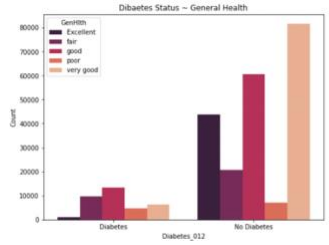
In [50]:
```
#group diabetes status & difficulty walking
dia_walk = eda2.groupby(['Diabetes_012', 'DiffWalk']).size().reset_index(name = 'Count')
print(dia_walk)

#visualize diabetes status - difficulty walking
plt.figure(figsize = (8,6))
sns.barplot(x = 'Diabetes_012', y = 'Count', hue = 'DiffWalk', data = dia_walk, palette="rocket")
plt.title("Dibaetes Status ~ Difficulty Walking")
plt.show()
```

```
  Diabetes_012 DiffWalk   Count
0     Diabetes       No   22225
1     Diabetes      yes   13121
2  No Diabetes       No  185434
3  No Diabetes      yes   28269
```



In [49]:
```
#group diabetes status & general health
diab_genhealth = eda2.groupby(['Diabetes_012', 'GenHlth']).size().reset_index(name = 'Count')
print(diab_genhealth)

#visualize diabetes status - general health
plt.figure(figsize = (8,6))
sns.barplot(x = 'Diabetes_012', y = 'Count', hue = 'GenHlth', data = diab_genhealth, palette="rocket")
plt.title("Dibaetes Status ~ General Health")
plt.show()
```

```
  Diabetes_012    GenHlth  Count
0     Diabetes  Excellent   1140
1     Diabetes       fair   9790
2     Diabetes       good  13457
3     Diabetes       poor   4578
4     Diabetes  very good   6381
5  No Diabetes  Excellent  43846
6  No Diabetes       fair  20755
7  No Diabetes       good  60461
8  No Diabetes       poor   7152
9  No Diabetes  very good  81489
```



Use of SMOTE function to balance dataset

In [51]:
```
from sklearn.model_selection import train_test_split
X = df2.toPandas().filter(items=['HighBP','HighChol','BMI','GenHlth','DiffWalk'])
Y = df2.toPandas().filter(items=['Diabetes_012'])

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.1, random_state=0)
```

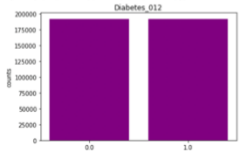
In [52]:
```
from imblearn.over_sampling import SMOTE
from imblearn.combine import SMOTEENN
from collections import Counter

sm = SMOTE(random_state=3)
x_train_res, y_train_res = sm.fit_resample(X_train, Y_train)
```

In [59]:
```
import pandas as pd
dataframe_1 = pd.DataFrame(x_train_res,columns=['HighBP','HighChol','BMI','GenHlth','DiffWalk'])
dataframe_2 = pd.DataFrame(y_train_res, columns = ['Diabetes_012'])
result = dataframe_1.combine_first(dataframe_2)
smote_1 = spark.createDataFrame(result)
```

In [60]:
```
import matplotlib.pyplot as plt
import numpy as np
responses = smote_1.groupBy('Diabetes_012').count().collect()
categories = [i[0] for i in responses]
counts = [i[1] for i in responses]
print(counts)
ind = np.array(range(len(categories)))
width = 0.35
plt.bar (ind, counts, color='purple')
plt.ylabel('counts' )
plt.title('Diabetes_012')
plt.xticks (ind, categories)
```

```
21/12/08 17:03:41 WARN TaskSetManager: Stage 128 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
[192322, 192322]
```

Out[60]:
```
([<matplotlib.axis.XTick at 0x7fbdfcb82970>,
  <matplotlib.axis.XTick at 0x7fbdfcb82850>],
 [Text(0, 0, '0.0'), Text(1, 0, '1.0')])
```



In [65]:
```
minmax_scaler = MinMaxScaler(inputCol="features", outputCol="scaled")
output1_data_smote = minmax_scaler.fit(output_data_smote).transform(output_data_smote)
output1_data_smote.show(5)
```

```
21/12/08 17:05:01 WARN TaskSetManager: Stage 144 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
```

```
+----+------------+--------+--------+------+--------+--------------------+--------------------+
| BMI|Diabetes_012|DiffWalk|GenHlth|HighBP|HighChol|            features|              scaled|
+----+------------+--------+--------+------+--------+--------------------+--------------------+
|24.0|         0.0|     0.0|     2.0|   1.0|     0.0|[1.0,0.0,24.0,2.0...|[1.0,0.0,0.139534...|
|25.0|         0.0|     0.0|     1.0|   0.0|     0.0|(5,[2,3],[25.0,1.0])|(5,[2],[0.1511627...|
|27.0|         0.0|     0.0|     2.0|   0.0|     0.0|(5,[2,3],[27.0,2.0])|(5,[2,3],[0.17441...|
|53.0|         0.0|     1.0|     3.0|   1.0|     0.0|[1.0,0.0,53.0,3.0...|[1.0,0.0,0.476744...|
|33.0|         0.0|     1.0|     3.0|   1.0|     1.0|[1.0,1.0,33.0,3.0...|[1.0,1.0,0.244186...|
+----+------------+--------+--------+------+--------+--------------------+--------------------+
only showing top 5 rows
```

```
21/12/08 17:05:03 WARN TaskSetManager: Stage 147 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
```

In [66]:
```
# Load training data
final_data = output1_data_smote.select('scaled','Diabetes_012')
final_data.show()
```

```
+--------------------+------------+
|              scaled|Diabetes_012|
+--------------------+------------+
|[1.0,0.0,0.139534...|         0.0|
|(5,[2],[0.1511627...|         0.0|
|(5,[2,3],[0.17441...|         0.0|
|[1.0,0.0,0.476744...|         0.0|
|[1.0,1.0,0.244186...|         0.0|
|[1.0,1.0,0.220930...|         0.0|
|(5,[2],[0.1627906...|         0.0|
|(5,[2],[0.2674418...|         0.0|
|(5,[2,3],[0.17441...|         0.0|
|[1.0,1.0,0.267441...|         1.0|
|[1.0,0.0,0.209302...|         0.0|
|[1.0,0.0,0.162790...|         0.0|
|[0.0,1.0,0.081395...|         0.0|
|(5,[2,3],[0.12790...|         0.0|
|[1.0,0.0,0.395348...|         0.0|
|(5,[2],[0.1744186...|         0.0|
|(5,[2,3],[0.23255...|         0.0|
|(5,[2,3],[0.09302...|         0.0|
|(5,[2],[0.2441860...|         0.0|
|[1.0,0.0,0.290697...|         0.0|
+--------------------+------------+
only showing top 20 rows
```

```
21/12/08 17:05:20 WARN TaskSetManager: Stage 148 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
```

In [62]:
```
print((smote_1.count(),len(imputeDF_1.columns)))
smote_1.groupby('Diabetes_012').count().show()
```

```
21/12/08 17:03:58 WARN TaskSetManager: Stage 134 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
(384644, 6)
21/12/08 17:03:58 WARN TaskSetManager: Stage 137 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
+-------------+------+
|Diabetes_012| count|
+-------------+------+
|          0.0|192322|
|          1.0|192322|
+-------------+------+
```

In [63]:
```
smote_1.show()
```

```
+----+-------------+---------+-------+------+---------+
| BMI|Diabetes_012|DiffWalk|GenHlth|HighBP|HighChol|
+----+-------------+---------+-------+------+---------+
|24.0|          0.0|      0.0|    2.0|   1.0|      0.0|
|25.0|          0.0|      0.0|    1.0|   0.0|      0.0|
|27.0|          0.0|      0.0|    2.0|   0.0|      0.0|
|53.0|          0.0|      1.0|    3.0|   1.0|      0.0|
|33.0|          0.0|      1.0|    3.0|   1.0|      1.0|
|31.0|          0.0|      0.0|    3.0|   1.0|      1.0|
|26.0|          0.0|      0.0|    1.0|   0.0|      0.0|
|35.0|          0.0|      0.0|    1.0|   0.0|      0.0|
|27.0|          0.0|      0.0|    3.0|   0.0|      0.0|
|35.0|          1.0|      0.0|    2.0|   1.0|      1.0|
|30.0|          0.0|      1.0|    3.0|   1.0|      0.0|
|26.0|          0.0|      0.0|    4.0|   1.0|      0.0|
|19.0|          0.0|      0.0|    2.0|   0.0|      1.0|
|23.0|          0.0|      0.0|    2.0|   0.0|      0.0|
|46.0|          0.0|      0.0|    3.0|   1.0|      0.0|
|27.0|          0.0|      0.0|    1.0|   0.0|      0.0|
|32.0|          0.0|      0.0|    2.0|   0.0|      0.0|
|20.0|          0.0|      0.0|    2.0|   0.0|      0.0|
|33.0|          0.0|      0.0|    1.0|   0.0|      0.0|
|37.0|          0.0|      0.0|    4.0|   1.0|      0.0|
+----+-------------+---------+-------+------+---------+
only showing top 20 rows
```

```
21/12/08 17:04:03 WARN TaskSetManager: Stage 140 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
```

In [64]:
```
from pyspark.ml.feature import VectorAssembler,MinMaxScaler
assembler_smote = VectorAssembler(inputCols=['HighBP','HighChol','BMI','GenHlth','DiffWalk'], outputCol='features')
output_data_smote = assembler_smote.transform(smote_1)
output_data_smote.groupby('Diabetes_012').count().show()
```

```
21/12/08 17:04:35 WARN TaskSetManager: Stage 141 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
+-------------+------+
|Diabetes_012| count|
+-------------+------+
|          0.0|192322|
|          1.0|192322|
+-------------+------+
```

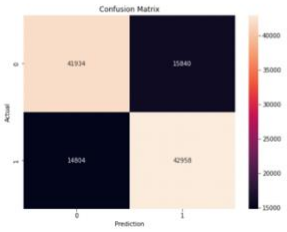
Logistic Regression

In [74]:
```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from sklearn.metrics import classification_report, confusion_matrix
train , test = final_data.randomSplit([0.7,0.3])
models = LogisticRegression(labelCol="Diabetes_012",featuresCol="scaled")
model = models.fit(train)
summary = model.summary
summary.predictions.describe().show()
predictions = model.evaluate(test)
predictions.predictions.show(20)
#evaluator = BinaryClassificationEvaluator(rawPredictionCol="rawPrediction", labelCol="Diabetes_012")
#evaluator.evaluate(model.transform(test))

y_true = predictions.predictions.select(['Diabetes_012']).collect()
y_pred = predictions.predictions.select(['prediction']).collect()

print(classification_report(y_true, y_pred))
plt.figure(figsize = (8,6))
sns.heatmap(confusion_matrix(y_true, y_pred), annot = True, fmt = ".0f")
plt.title("Confusion Matrix")
plt.xlabel("Prediction")
plt.ylabel("Actual")
plt.show()
```

```
21/12/08 17:29:15 WARN TaskSetManager: Stage 1226 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:29:17 WARN TaskSetManager: Stage 1228 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:29:17 WARN TaskSetManager: Stage 1230 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:29:18 WARN TaskSetManager: Stage 1232 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:29:18 WARN TaskSetManager: Stage 1234 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:29:18 WARN TaskSetManager: Stage 1236 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:29:18 WARN TaskSetManager: Stage 1238 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:29:18 WARN TaskSetManager: Stage 1240 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:29:18 WARN TaskSetManager: Stage 1242 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:29:18 WARN TaskSetManager: Stage 1244 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:29:18 WARN TaskSetManager: Stage 1246 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
/Users/rishisingh/.local/share/virtualenvs/spark_test-2CXULP8/lib/python3.9/site-packages/pyspark/sql/context.py:125: FutureWarning: Deprecated in 3.0.0. Use SparkSession.builder.getOrCreate() instead.
  warnings.warn(
21/12/08 17:29:19 WARN TaskSetManager: Stage 1248 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
```

```
+-------+------------------+------------------+
|summary|      Diabetes_012|        prediction|
+-------+------------------+------------------+
|  count|            269108|            269108|
|   mean|0.5000222958812075|0.5102969811376845|
| stddev|0.5000009285005321|0.4998948897357867|
|    min|               0.0|               0.0|
|    max|               1.0|               1.0|
+-------+------------------+------------------+
```

```
              precision    recall  f1-score   support

         0.0       0.74      0.73      0.73     57774
         1.0       0.73      0.74      0.74     57762

    accuracy                           0.73    115536
   macro avg       0.73      0.73      0.73    115536
weighted avg       0.73      0.73      0.73    115536
```



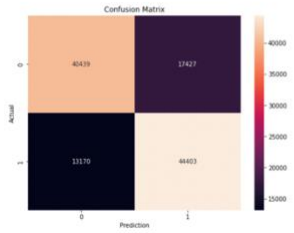
Random Forest Classifer

In [75]:
```
from pyspark.ml.classification import RandomForestClassifier

train , test = final_data.randomSplit([0.7,0.3])
model_rf = RandomForestClassifier(labelCol='Diabetes_012',featuresCol="scaled")
model_rf = model_rf.fit(train)
summary = model_rf.summary
summary.predictions.describe().show()
predictions = model_rf.evaluate(test)
predictions.predictions.show(20)
#evaluator = BinaryClassificationEvaluator(rawPredictionCol="rawPrediction", labelCol="Diabetes_012")
#evaluator.evaluate(model.transform(test))

y_true = predictions.predictions.select(['Diabetes_012']).collect()
y_pred = predictions.predictions.select(['prediction']).collect()

print(classification_report(y_true, y_pred))
plt.figure(figsize = (8,6))
sns.heatmap(confusion_matrix(y_true, y_pred), annot = True, fmt = ".0f")
plt.title("Confusion Matrix")
plt.xlabel("Prediction")
plt.ylabel("Actual")
plt.show()
```

```
21/12/08 17:29:30 WARN TaskSetManager: Stage 1254 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:29:31 WARN TaskSetManager: Stage 1257 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:29:32 WARN TaskSetManager: Stage 1258 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:29:33 WARN TaskSetManager: Stage 1259 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:29:34 WARN TaskSetManager: Stage 1261 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:29:36 WARN TaskSetManager: Stage 1263 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:29:36 WARN TaskSetManager: Stage 1265 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:29:36 WARN TaskSetManager: Stage 1267 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:29:37 WARN TaskSetManager: Stage 1269 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
/Users/rishisingh/.local/share/virtualenvs/spark_test-2CKUZILP8/lib/python3.9/site-packages/pyspark/sql/context.py:125: FutureWarning: Deprecated in 3.0.0. Use SparkSession.builder.getOrCreate() instead.
  warnings.warn(
21/12/08 17:29:38 WARN TaskSetManager: Stage 1271 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
```

```
+-------+-----------------+------------------+
|summary|     Diabetes_012|        prediction|
+-------+-----------------+------------------+
|  count|           269205|            269205|
|   mean|0.5005441949443733| 0.5359187236492635|
| stddev|0.5000006325141266|0.49870910274697416|
|    min|              0.0|               0.0|
|    max|              1.0|               1.0|
+-------+-----------------+------------------+
```

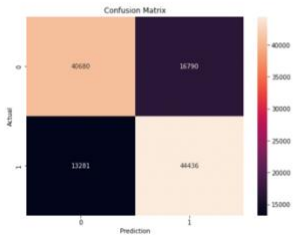
	precision	recall	f1-score	support
0.0	0.75	0.70	0.73	57866
1.0	0.72	0.77	0.74	57573
accuracy			0.73	115439
macro avg	0.74	0.74	0.73	115439
weighted avg	0.74	0.73	0.73	115439



Gradient Boosting Tree

In [73]:
```
from pyspark.ml.classification import GBTClassifier

train , test = final_data.randomSplit([0.7,0.3])

models_gbt = GBTClassifier(labelCol='Diabetes_012',featuresCol="scaled")
model_gbt = models_gbt.fit(train)
predictions = model_gbt.transform(test)
predictions.show(20)
#evaluator = BinaryClassificationEvaluator(rawPredictionCol="rawPrediction", labelCol="Diabetes_012")
#evaluator.evaluate(model.transform(test))

y_true = predictions.select(['Diabetes_012']).collect()
y_pred = predictions.select(['prediction']).collect()

print(classification_report(y_true, y_pred))
plt.figure(figsize = (8,6))
sns.heatmap(confusion_matrix(y_true, y_pred), annot = True, fmt = ".0f")
plt.title("Confusion Matrix")
plt.xlabel("Prediction")
plt.ylabel("Actual")
plt.show()
```

	precision	recall	f1-score	support
0.0	0.75	0.71	0.73	57470
1.0	0.73	0.77	0.75	57717
accuracy			0.74	115187
macro avg	0.74	0.74	0.74	115187
weighted avg	0.74	0.74	0.74	115187



Evaluation of performance runtime

Logistic Regression runtime evaluation

In [77]:

```
import time
start = time.time()

train , test = final_data.randomSplit([0.7,0.3])
models = LogisticRegression(labelCol='Diabetes_012',featuresCol="scaled")
model = models.fit(train)
summary = model.summary
summary.predictions.describe().show()
predictions = model.evaluate(test)
end = time.time()
predictions.predictions.show(20)

timetaken = end - start
print("Time taken to run the Linear Regression model:", timetaken )
```

```
21/12/08 17:34:08 WARN TaskSetManager: Stage 1303 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:34:09 WARN TaskSetManager: Stage 1305 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:34:11 WARN TaskSetManager: Stage 1307 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:34:11 WARN TaskSetManager: Stage 1309 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:34:11 WARN TaskSetManager: Stage 1311 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:34:11 WARN TaskSetManager: Stage 1313 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:34:11 WARN TaskSetManager: Stage 1315 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:34:12 WARN TaskSetManager: Stage 1317 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:34:12 WARN TaskSetManager: Stage 1319 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:34:12 WARN TaskSetManager: Stage 1321 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:34:12 WARN TaskSetManager: Stage 1323 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:34:13 WARN TaskSetManager: Stage 1325 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
```

summary	Diabetes_012	prediction
count	268967	268967
mean	0.500150576092978	0.5090736038250046
stddev	0.5000009068114447	0.4999185922658198
min	0.0	0.0
max	1.0	1.0

scaled	Diabetes_012	rawPrediction	probability	prediction
(5,[0,2],[1.0,0.0...	0.0	[1.92375382331038...	[0.87255644929344...	0.0
(5,[0,2],[1.0,0.0...	0.0	[1.80072040788416...	[0.85823660725161...	0.0
(5,[0,2],[1.0,0.0...	0.0	[1.80072040788416...	[0.85823660725161...	0.0
(5,[0,2],[1.0,0.0...	0.0	[1.73920370017106...	[0.85058589234649...	0.0
(5,[0,2],[1.0,0.0...	0.0	[1.73920370017106...	[0.85058589234649...	0.0
(5,[0,2],[1.0,0.0...	0.0	[1.67768699245795...	[0.84259800781226...	0.0
(5,[0,2],[1.0,0.0...	0.0	[1.67768699245795...	[0.84259800781226...	0.0
(5,[0,2],[1.0,0.0...	0.0	[1.67768699245795...	[0.84259800781226...	0.0
(5,[0,2],[1.0,0.1...	0.0	[1.61617028474484...	[0.83426628785360...	0.0
(5,[0,2],[1.0,0.1...	0.0	[1.61617028474484...	[0.83426628785360...	0.0
(5,[0,2],[1.0,0.1...	0.0	[1.61617028474484...	[0.83426628785360...	0.0
(5,[0,2],[1.0,0.1...	0.0	[1.61617028474484...	[0.83426628785360...	0.0
(5,[0,2],[1.0,0.1...	0.0	[1.61617028474484...	[0.83426628785360...	0.0
(5,[0,2],[1.0,0.1...	0.0	[1.61617028474484...	[0.83426628785360...	0.0
(5,[0,2],[1.0,0.1...	0.0	[1.55465357703173...	[0.82558483700183...	0.0
(5,[0,2],[1.0,0.1...	0.0	[1.55465357703173...	[0.82558483700183...	0.0
(5,[0,2],[1.0,0.1...	0.0	[1.55465357703173...	[0.82558483700183...	0.0
(5,[0,2],[1.0,0.1...	0.0	[1.55465357703173...	[0.82558483700183...	0.0
(5,[0,2],[1.0,0.1...	0.0	[1.55465357703173...	[0.82558483700183...	0.0
(5,[0,2],[1.0,0.1...	0.0	[1.55465357703173...	[0.82558483700183...	0.0

```
only showing top 20 rows

Time taken to run the Linear Regression model: 6.0564799308776855
```

Random Forest runtime evaluation

In [81]:

```python
import time
start = time.time()

train , test = final_data.randomSplit([0.7,0.3])
models = RandomForestClassifier(labelCol='Diabetes_012',featuresCol="scaled")
model = models.fit(train)
summary = model.summary
summary.predictions.describe().show()
predictions = model.evaluate(test)
end = time.time()
predictions.predictions.show(20)

timetaken = end - start
print("Time taken to run the Random Forest model:", timetaken )
```

```
21/12/08 17:36:16 WARN TaskSetManager: Stage 1350 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:36:17 WARN TaskSetManager: Stage 1353 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:36:17 WARN TaskSetManager: Stage 1354 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:36:18 WARN TaskSetManager: Stage 1355 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:36:19 WARN TaskSetManager: Stage 1357 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:36:20 WARN TaskSetManager: Stage 1359 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:36:21 WARN TaskSetManager: Stage 1361 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:36:21 WARN TaskSetManager: Stage 1363 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:36:21 WARN TaskSetManager: Stage 1365 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
21/12/08 17:36:22 WARN TaskSetManager: Stage 1367 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
```

```
+-------+-------------------+-------------------+
|summary|       Diabetes_012|         prediction|
+-------+-------------------+-------------------+
|  count|             269304|             269304|
|   mean|0.49955440691560465|  0.546148590440543|
| stddev| 0.5000007297679064|0.49786667694731357|
|    min|                0.0|                0.0|
|    max|                1.0|                1.0|
+-------+-------------------+-------------------+
```

```
21/12/08 17:36:24 WARN TaskSetManager: Stage 1370 contains a task of very large size (2684 KiB). The maximum recommended task size is 1000 KiB.
```

```
+--------------------+------------+--------------------+--------------------+----------+
|              scaled|Diabetes_012|       rawPrediction|         probability|prediction|
+--------------------+------------+--------------------+--------------------+----------+
|(5,[0,2],[1.0,0.0...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
|(5,[0,2],[1.0,0.0...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
|(5,[0,2],[1.0,0.0...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
|(5,[0,2],[1.0,0.0...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
|(5,[0,2],[1.0,0.0...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
|(5,[0,2],[1.0,0.0...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
|(5,[0,2],[1.0,0.0...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
|(5,[0,2],[1.0,0.0...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
|(5,[0,2],[1.0,0.0...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
|(5,[0,2],[1.0,0.1...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
|(5,[0,2],[1.0,0.1...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
|(5,[0,2],[1.0,0.1...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
|(5,[0,2],[1.0,0.1...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
|(5,[0,2],[1.0,0.1...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
|(5,[0,2],[1.0,0.1...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
|(5,[0,2],[1.0,0.1...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
|(5,[0,2],[1.0,0.1...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
|(5,[0,2],[1.0,0.1...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
|(5,[0,2],[1.0,0.1...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
|(5,[0,2],[1.0,0.1...|         0.0|[13.0748788727381...|[0.65374394363690...|       0.0|
+--------------------+------------+--------------------+--------------------+----------+
only showing top 20 rows

Time taken to run the Random Forest model: 7.963494777679443
```

Scalability Evaluation

In [86]:

```python
def runtime(df,model_user):
    lr = LogisticRegression(labelCol='Diabetes_012',featuresCol="scaled")
    rfc = RandomForestClassifier(labelCol='Diabetes_012',featuresCol="scaled")
    gbt = GBTClassifier(labelCol='Diabetes_012',featuresCol="scaled")

    sampling = [0.25,0.5,0.75,1]
    timeEnd = []
    time = []
    if model_user == "lr":
        for sample in sampling:
            start = timer()
            final_data = output1_data_smote.select('scaled','Diabetes_012')
            final_data.show()
            train , test = final_data.randomSplit([0.7,0.3])
            t = train.count() * sample
            train = train.limit(int(t))
            models = lr
            model = models.fit(train)
            #predictions = model.evaluate(test)
            predictions = model.transform(test)
            end = timer()
            final = end-start
            time.append(final)
            timeEnd.append(end)
    elif model_user == "rfc":
        for sample in sampling:
            start = timer()
            final_data = output1_data_smote.select('scaled','Diabetes_012')
            final_data.show()
            train , test = final_data.randomSplit([0.7,0.3])
            t = train.count() * sample
            train = train.limit(int(t))
            models = rfc
            model = models.fit(train)
            #predictions = model.evaluate(test)
            predictions = model.transform(test)
            end = timer()
            final = end-start
            time.append(final)
            timeEnd.append(end)
    elif model_user == "gbt":
        for sample in sampling:
            start = timer()
            final_data = output1_data_smote.select('scaled','Diabetes_012')
            final_data.show()
            train , test = final_data.randomSplit([0.7,0.3])
            t = train.count() * sample
            train = train.limit(int(t))
            models = lr
            model = models.fit(train)
            #predictions = model.evaluate(test)
            predictions = model.transform(test)
            end = timer()
            final = end-start
            time.append(final)
            timeEnd.append(end)


    return sampling, time
```

In [91]:

```python
from timeit import default_timer as timer
sample, time = runtime(final_data,model_user="lr")
print(sample)
print(time)
```