VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Belagavi – 590018, Karnataka State, India

PROJECT ENTITLED

"Diabetes Prediction Using PySpark"

Submitted in partial fulfilment of the requirements for the award of degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

For the academic year 2021-2022

Submitted by:

SUJITH N E	**(1MV18CS113)**
TEJA E	**(1MV18CS117)**
YALAMANCHILI SAI GOKUL	**(1MV18CS127)**

Mini Project Carried out at

Sir M. Visvesvaraya Institute of Technology

Bengaluru - 562157

Under Guidance of

Dr. PALLAVI R

Associate Professor

SIR M. VISVESVARAYA INSTITUTE OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
HUNASAMARANAHALLI BENGALURU – 56215

SIR M. VISVESVARAYA INSTITUTE OF TECHNOLOGY

Krishnadevaraya Nagar, International Airport Road,
Hunasmaranahalli, Bengaluru – 562157

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

It is certified that the project work entitled " **Diabetes Prediction Using PySpark** " is carried out by **SUJITH N E (1MV18CS113), TEJA E (1MV18CS117), YALAMANCHILI SAI GOKUL (1MV18CS127)** bonafide students of **Sir M Visvesvaraya Institute of Technology** in partialfulfilment for the award of the Degree of Bachelor of Engineering in Computer Science and Engineering of the **Visvesvaraya Technological University, Belagavi** during the year **2021- 2022**. It is certified that all corrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the course of Bachelor of Engineering.

Name & Signature of Guide	Name & Signature of HOD	Name & Signature of Principal
Dr. Pallavi R	**Dr. G. C. Bhanu Prakash**	**Dr. V.R. Manjunath**
Assoc. Prof	HOD, Dept. Of CSE,	Principal,
Dept. Of CSE, Sir MVIT	Sir MVIT	Sir MVIT
Bengaluru - 562157	Bengaluru - 562157	Bengaluru - 562157

DECLARATION

We hereby declare that the mini project embodied in this dissertation has been carried out by us and no part has been submitted for any degree or diploma of any institution previously.

Place: Bengaluru

Date:

Signature of Students:

Sujith N E
(1MV18CS113)

Teja E
(1MV18CS117)

Yalamanchili Sai
Gokul
(1MV18CS127)

ABSTRACT

Diabetes is an illness caused because of high glucose level in a human body. Diabetes should not be ignored if it is untreated then Diabetes may cause some major issues in a person like: heart related problems, kidney problem, blood pressure, eye damage and it can also affects other organs of human body. Diabetes can be controlled if it is predicted earlier. To achieve this goal this project work we will do early prediction of Diabetes in a human body or a patient for a higher accuracy through applying, Various Machine Learning Techniques. Machine learning techniques Provide better result for prediction by con- structing models from datasets collected from patients.

In this work we will use Machine Learning Correlation Analysis and Feature extraction techniques on a dataset to predict diabetes. Correlation is a statistical measure that indicates the extent to which two or more variables fluctuate together. In simple terms, it tells us how much does one variable changes for a slight change in another variable. It may take positive, negative and zero values depending on the direction of the change. A high correlation value between a dependent variable and an independent variable indicates that the independent variable is of very high significance in determining the output. Feature extraction is a part of the dimensionality reduction process, in which, an initial set of the raw data is divided and reduced to more manageable groups. So, when you want to process it will be easier. The most important characteristic of these large data sets is that they have a large number of variables. These variables require a lot of computing resources to process. So, Feature extraction helps to get the best feature from those big data sets by selecting and combining variables into features, thus, effectively reducing the amount of data. These features are easy to process, but still able to describe the actual data set with accuracy and originality.

TABLE OF CONTENTS

INTRODUCTION

Diabetes is noxious diseases in the world. Diabetes caused because of obesity or high blood glucose level, and so forth. It affects the hormone insulin, resulting in abnormal metabolism of crabs and improves level of sugar in the blood. Diabetes occurs when body does not make enough insulin. According to (WHO) World Health Organization about 422 million people suffering from diabetes particularly from low or idle income countries. And this could be increased to 490 billion up to the year of 2030. However prevalence of diabetes is found among various Countries like Canada, China, and India etc. Population of India is now more than 100 million so the actual number of diabetics in India is 40 million. Diabetes is major cause of death in the world. Early prediction of disease like diabetes can be controlled and save the human life. To accomplish this, this work explores prediction of diabetes by taking various attributes related to diabetes disease. For this purpose we use the Pima Indian Diabetes Dataset, we apply various Machine Learning Correlation and feature Extraction Techniques to predict diabetes. Various Machine Learning Techniques provide efficient result to collect Knowledge by building various classification and ensemble models from collected dataset. Such collected data can be useful to predict diabetes. Various techniques of Machine Learning can capable to do prediction, however it’s tough to choose best technique. Thus for this purpose we apply popular Correlation and Feature Extraction methods on dataset for prediction.

METHODOLOGY

Goal of our project is to investigate for model to predict diabetes with better accuracy. We used Correlation Analysis and feature extraction algorithms to predict diabetes.

A) Dataset Description : the data is gathered from UCI repository which is named as Pima Indian Diabetes Dataset. The dataset have many attributes of 2000 patients.

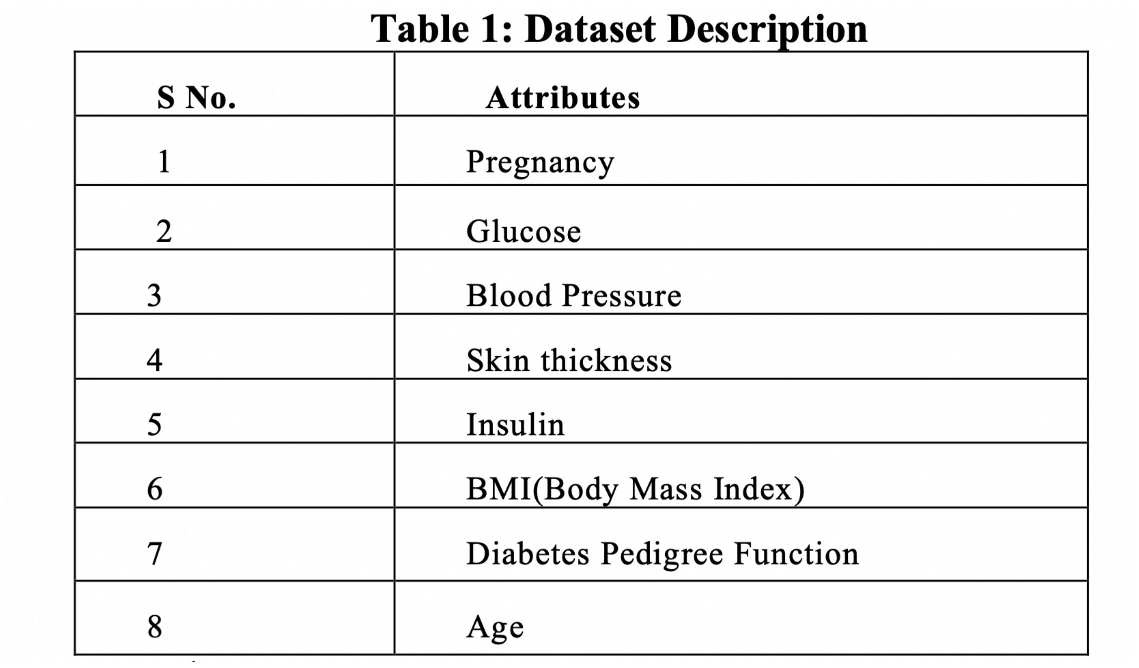


Table 1: Dataset Description

S No.	Attributes
1	Pregnancy
2	Glucose
3	Blood Pressure
4	Skin thickness
5	Insulin
6	BMI(Body Mass Index)
7	Diabetes Pedigree Function
8	Age

The 9th attribute is class variable of each data points. This class variable shows the outcome 0 and 1 for diabetics which indicates positive or negative for diabetics.

B) Data Preprocessing : Data preprocessing is most important process. Mostly healthcare related data contains missing value and other impurities that can cause effectiveness of data. To improve quality and effectiveness obtained after mining process, Data preprocessing is done. To use Machine Learning Techniques on the dataset effectively this process is essential for accurate result and successful prediction. For Pima Indian diabetes dataset we need to perform pre processing in two steps.

1). **Missing Values removal-** Remove all the instances that have zero (0) as worth. Having zero as worth is not possible. Therefore this instance is eliminated. Through eliminating irrelevant features/instances we make feature subset and this process is called features subset selection, which reduces dimensionality of data and help to work faster.

2). **Splitting of data-** After cleaning the data, data is normalized in training and testing the model. When data is spitted then we train algorithm on the training data set and keep test data set aside. This training process will produce the training model based on logic and algorithms and values of the feature in training data. Basically aim of normalization is to bring all the attributes under same scale.

C) Apply Machine Learning : When data has been ready we apply Machine Learning Technique. The methods applied on Pima Indians diabetes dataset are Correlation analysis and feature extraction techniques, to predict diabetes.
The Techniques are follows-

Correlation Analysis : Correlation is a statistical measure that indicates the extent to which two or more variables fluctuate together. In simple terms, it tells us how much does one variable changes for a slight change in another variable. It may take positive, negative and zero values depending on the direction of the change. A high correlation value between a dependent variable and an independent variable indicates that the independent variable is of very high significance in determining the output. In a multiple regression setup where there are many factors, it is imperative to find the correlation between the dependent and all the independent variables to build a more viable model with higher accuracy. One must always remember that more number of features does not imply better accuracy. More features may lead to a decline in the accuracy if they contain any irrelevant features creating unrequired noise in our model.

Feature Extraction : Feature Extraction aims to reduce the number of features in a dataset by creating new features from the existing ones (and then discarding the original features). These new reduced set of features should then be able to summarize most of the information contained in the original set of features. In this way, a summarised version of the original features can be created from a combination of the original set.

IMPLEMENTATION

A) PYTHON :

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

B) JUPYTER NOTEBOOK :

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.
Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

C) PYSPARK :

Apache Spark is written in Scala programming language. PySpark has been released in order to support the collaboration of Apache Spark and Python, it actually is a Python API for Spark. In addition, PySpark, helps you interface with Resilient Distributed Datasets (RDDs) in Apache Spark and Python programming language. This has been achieved by taking advantage of the Py4j library.Py4J is a popular library which is integrated within PySpark and allows python to dynamically interface with JVM objects. PySpark features quite a few libraries for writing efficient programs. Furthermore, there are various external libraries that are also compatible.

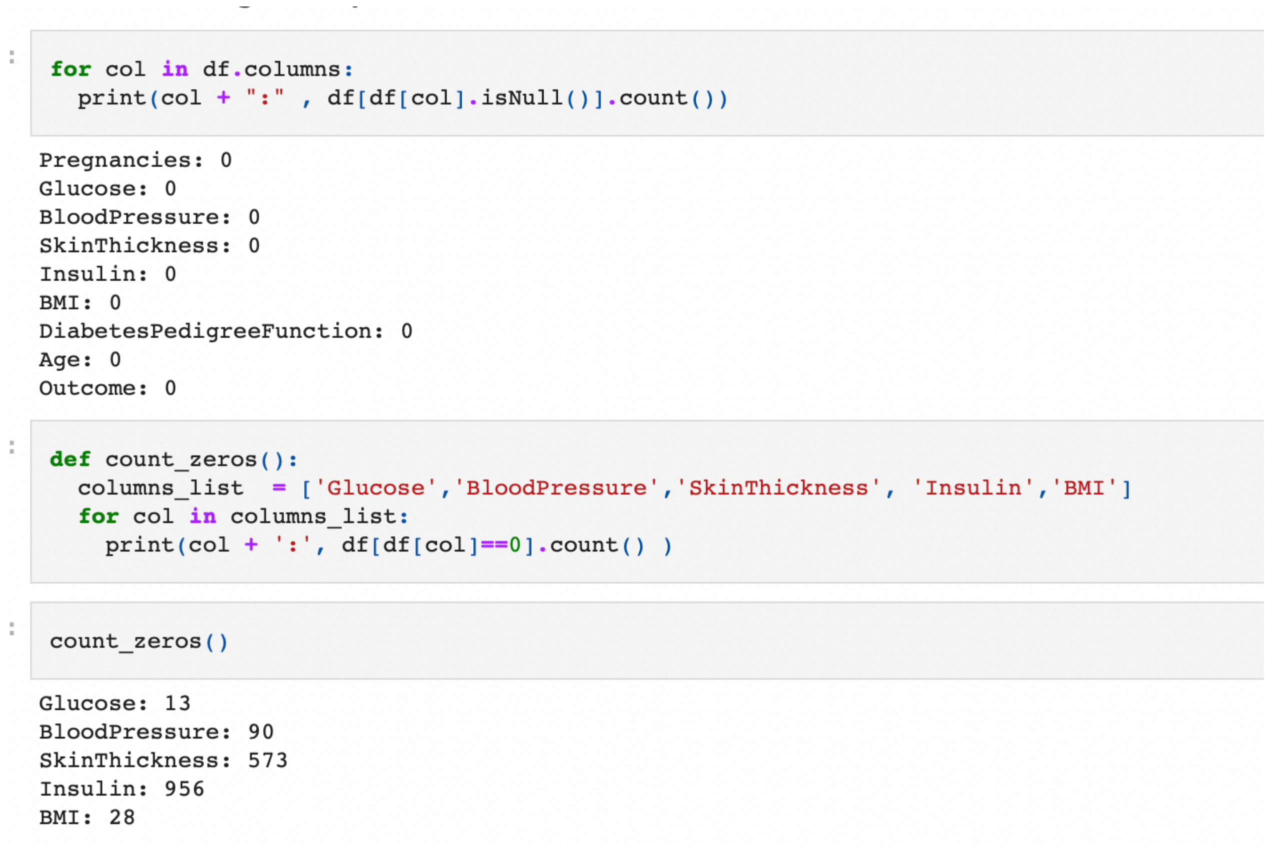
PySparkSQL :
A PySpark library to apply SQL-like analysis on a huge amount of structured or semi-structured data. We can also use SQL queries with PySparkSQL. It can also be connected to Apache Hive. HiveQL can be also be applied. PySparkSQL is a wrapper over the PySpark core. PySparkSQL introduced the DataFrame, a tabular representation of structured data that is similar to that of a table from a relational database management system.

Logistic Regression :
Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes. In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no). Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

Binary Classification Evaluator:
Evaluator for binary classification, which expects input columns rawPrediction, label and an optional weight column. The rawPrediction column can be of type double (binary 0/1 prediction, or probability of label 1) or of type vector (length-2 vector of raw predictions, scores, or label probabilities).

Data Cleaning and Preparation :

Missing values and other impurities are common in healthcare data, which can reduce its effectiveness. Data preprocessing is done to increase the quality and effectiveness of the results obtained after the mining process. This method is necessary for accurate results and good prediction when using Machine Learning Techniques on a dataset.



```python
for col in df.columns:
  print(col + ":" , df[df[col].isNull()].count())
```

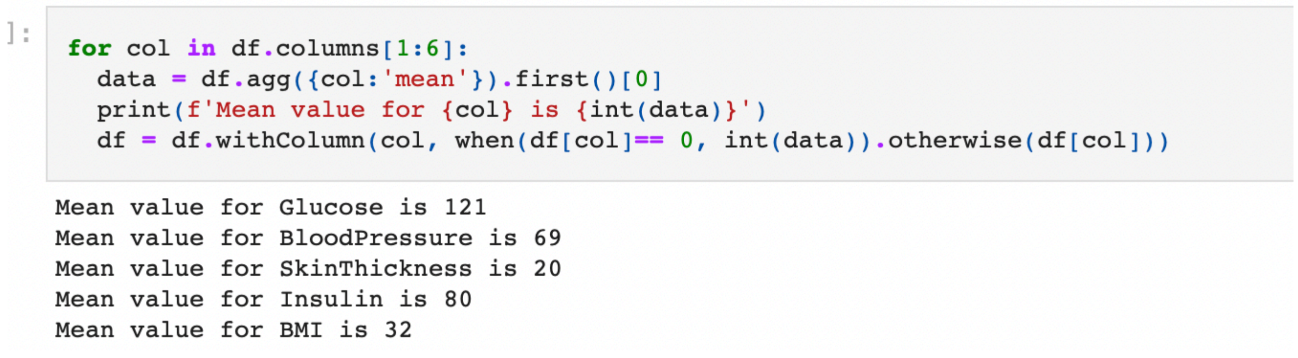
```
Pregnancies: 0
Glucose: 0
BloodPressure: 0
SkinThickness: 0
Insulin: 0
BMI: 0
DiabetesPedigreeFunction: 0
Age: 0
Outcome: 0
```

```python
def count_zeros():
  columns_list  = ['Glucose','BloodPressure','SkinThickness', 'Insulin','BMI']
  for col in columns_list:
    print(col + ':', df[df[col]==0].count() )
```

```python
count_zeros()
```

```
Glucose: 13
BloodPressure: 90
SkinThickness: 573
Insulin: 956
BMI: 28
```

Finding Mean Values :

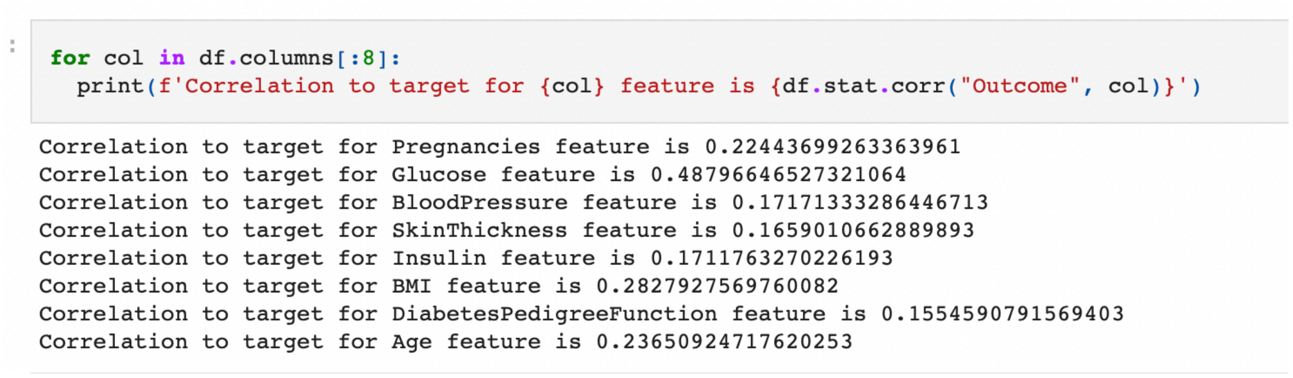


```python
for col in df.columns[1:6]:
  data = df.agg({col:'mean'}).first()[0]
  print(f'Mean value for {col} is {int(data)}')
  df = df.withColumn(col, when(df[col]== 0, int(data)).otherwise(df[col]))
```

```
Mean value for Glucose is 121
Mean value for BloodPressure is 69
Mean value for SkinThickness is 20
Mean value for Insulin is 80
Mean value for BMI is 32
```

Correlation Analysis :

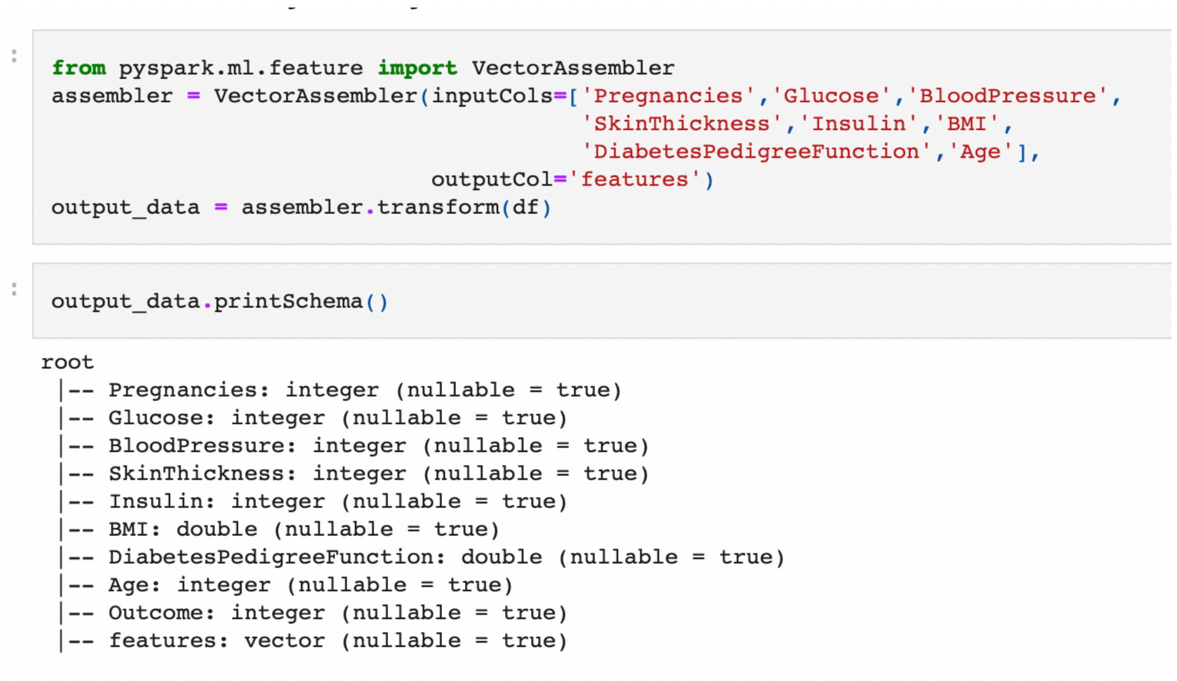
Correlation is a statistical measure that determines how closely two or more variables fluctuate. In simple terms, it informs us how much one variable varies when another variable is changed slightly. Depending on the direction of the shift, it might take positive, negative, or zero values.



```
for col in df.columns[:8]:
  print(f'Correlation to target for {col} feature is {df.stat.corr("Outcome", col)}')
```

```
Correlation to target for Pregnancies feature is 0.22443699263363961
Correlation to target for Glucose feature is 0.48796646527321064
Correlation to target for BloodPressure feature is 0.17171333286446713
Correlation to target for SkinThickness feature is 0.1659010662889893
Correlation to target for Insulin feature is 0.1711763270226193
Correlation to target for BMI feature is 0.2827927569760082
Correlation to target for DiabetesPedigreeFunction feature is 0.1554590791569403
Correlation to target for Age feature is 0.23650924717620253
```

Feature Selection :



```
from pyspark.ml.feature import VectorAssembler
assembler = VectorAssembler(inputCols=['Pregnancies','Glucose','BloodPressure',
                                       'SkinThickness','Insulin','BMI',
                                       'DiabetesPedigreeFunction','Age'],
                            outputCol='features')
output_data = assembler.transform(df)
```

```
output_data.printSchema()
```

```
root
 |-- Pregnancies: integer (nullable = true)
 |-- Glucose: integer (nullable = true)
 |-- BloodPressure: integer (nullable = true)
 |-- SkinThickness: integer (nullable = true)
 |-- Insulin: integer (nullable = true)
 |-- BMI: double (nullable = true)
 |-- DiabetesPedigreeFunction: double (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Outcome: integer (nullable = true)
 |-- features: vector (nullable = true)
```

In [1]:
```
! pip install pyspark
```

```
Requirement already satisfied: pyspark in ./opt/anaconda3/lib/pyth
on3.8/site-packages (3.0.1)
Requirement already satisfied: py4j==0.10.9 in ./opt/anaconda3/lib
/python3.8/site-packages (from pyspark) (0.10.9)
```

In [2]:
```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('spark').getOrCreate()
```

In [3]:
```
df = spark.read.csv('diabetes.csv', header=True, inferSchema=True)
```

In [39]:
```
df.show(10)
```

```
+-----------+-------+-------------+-------------+-------+----+----
--------------------+---+-------+
|Pregnancies|Glucose|BloodPressure|SkinThickness|Insulin| BMI|Diab
etesPedigreeFunction|Age|Outcome|
+-----------+-------+-------------+-------------+-------+----+----
--------------------+---+-------+
|          2|    138|           62|           35|     80|33.6|
0.127| 47|      1|
|          0|     84|           82|           31|    125|38.2|
0.233| 23|      0|
|          0|    145|           69|           20|     80|44.2|
0.63| 31|      1|
|          0|    135|           68|           42|    250|42.3|
0.365| 24|      1|
|          1|    139|           62|           41|    480|40.7|
0.536| 21|      0|
|          0|    173|           78|           32|    265|46.5|
1.159| 58|      0|
|          4|     99|           72|           17|     80|25.6|
0.294| 28|      0|
|          8|    194|           80|           20|     80|26.1|
0.551| 67|      0|
|          2|     83|           65|           28|     66|36.8|
0.629| 24|      0|
|          2|     89|           90|           30|     80|33.5|
0.292| 42|      0|
+-----------+-------+-------------+-------------+-------+----+----
--------------------+---+-------+
only showing top 10 rows
```

In [5]:
```
df.printSchema()
```

```
root
 |-- Pregnancies: integer (nullable = true)
 |-- Glucose: integer (nullable = true)
 |-- BloodPressure: integer (nullable = true)
 |-- SkinThickness: integer (nullable = true)
 |-- Insulin: integer (nullable = true)
 |-- BMI: double (nullable = true)
 |-- DiabetesPedigreeFunction: double (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Outcome: integer (nullable = true)
```

In [6]:
```
print((df.count(), len(df.columns)))
```

```
(2000, 9)
```

In [7]:
```
df.groupby('Outcome').count().show()
```

```
+-------+-----+
|Outcome|count|
+-------+-----+
|      1|  684|
|      0| 1316|
+-------+-----+
```

In [8]:
```
df.describe().show()
```

```
+-------+------------------+------------------+------------------+-
----------------+-----------------+------------------+------------
------------+------------------+------------------+
|summary|       Pregnancies|           Glucose|     BloodPressure|
SkinThickness|           Insulin|               BMI|DiabetesPedigre
eFunction|                Age|           Outcome|
+-------+------------------+------------------+------------------+-
----------------+-----------------+------------------+------------
------------+------------------+------------------+
|  count|              2000|              2000|              2000|
2000|              2000|              2000|                  2000
|              2000|              2000|
|   mean|            3.7035|          121.1825|           69.1455|
20.935|            80.254|32.192999999999984|     0.470929999999999
74|           33.0905|             0.342|
| stddev|3.306063032730656|32.068635649902916|19.188314815604098|1
6.10324290992682|111.1805335457595| 8.149900701279762|      0.3235
525586811429|11.786423106049496|0.4744982342297426|
|    min|                 0|                 0|                 0|
0|                 0|               0.0|                 0.078|
21|                 0|
|    max|                17|               199|               122|
110|               744|              80.6|                  2.42|
81|                 1|
+-------+------------------+------------------+------------------+-
----------------+-----------------+------------------+------------
------------+------------------+------------------+
```

In [9]:
```
for col in df.columns:
  print(col + ":" , df[df[col].isNull()].count())
```

```
Pregnancies: 0
Glucose: 0
BloodPressure: 0
SkinThickness: 0
Insulin: 0
BMI: 0
DiabetesPedigreeFunction: 0
Age: 0
Outcome: 0
```

In [10]:
```
def count_zeros():
  columns_list  = ['Glucose','BloodPressure','SkinThickness', 'Insu
  for col in columns_list:
    print(col + ':', df[df[col]==0].count() )
```

In [11]:
```
count_zeros()
```

```
Glucose: 13
BloodPressure: 90
SkinThickness: 573
Insulin: 956
BMI: 28
```

In [12]:
```
from pyspark.sql.functions import *
```

In [13]:
```
df.agg({'BMI':'mean'}).first()[0]
```

Out[13]:
```
32.192999999999984
```

In [14]:
```
for col in df.columns[1:6]:
  data = df.agg({col:'mean'}).first()[0]
  print(f'Mean value for {col} is {int(data)}')
  df = df.withColumn(col, when(df[col]== 0, int(data)).otherwise(df
```

```
Mean value for Glucose is 121
Mean value for BloodPressure is 69
Mean value for SkinThickness is 20
Mean value for Insulin is 80
Mean value for BMI is 32
```

In [40]:
```
df.show(10)
```

```
+-----------+-------+-------------+-------------+-------+----+----
--------------------+---+-------+
|Pregnancies|Glucose|BloodPressure|SkinThickness|Insulin| BMI|Diab
etesPedigreeFunction|Age|Outcome|
+-----------+-------+-------------+-------------+-------+----+----
--------------------+---+-------+
|          2|    138|           62|           35|     80|33.6|
0.127| 47|      1|
|          0|     84|           82|           31|    125|38.2|
0.233| 23|      0|
|          0|    145|           69|           20|     80|44.2|
0.63| 31|      1|
|          0|    135|           68|           42|    250|42.3|
0.365| 24|      1|
|          1|    139|           62|           41|    480|40.7|
0.536| 21|      0|
|          0|    173|           78|           32|    265|46.5|
1.159| 58|      0|
|          4|     99|           72|           17|     80|25.6|
0.294| 28|      0|
|          8|    194|           80|           20|     80|26.1|
0.551| 67|      0|
|          2|     83|           65|           28|     66|36.8|
0.629| 24|      0|
|          2|     89|           90|           30|     80|33.5|
0.292| 42|      0|
+-----------+-------+-------------+-------------+-------+----+----
--------------------+---+-------+
only showing top 10 rows
```

In [16]:
```
for col in df.columns[:8]:
  print(f'Correlation to target for {col} feature is {df.stat.corr(
```

```
Correlation to target for Pregnancies feature is 0.224436992633639
61
Correlation to target for Glucose feature is 0.48796646527321064
Correlation to target for BloodPressure feature is 0.1717133328644
6713
Correlation to target for SkinThickness feature is 0.1659010662889
893
Correlation to target for Insulin feature is 0.1711763270226193
Correlation to target for BMI feature is 0.2827927569760082
Correlation to target for DiabetesPedigreeFunction feature is 0.15
54590791569403
Correlation to target for Age feature is 0.23650924717620253
```

In [17]:
```
from pyspark.ml.feature import VectorAssembler
assembler = VectorAssembler(inputCols=['Pregnancies','Glucose','Blo
                                       'SkinThickness','Insulin','B
                                       'DiabetesPedigreeFunction','
                            outputCol='features')
output_data = assembler.transform(df)
```

In [18]:
```
output_data.printSchema()
```

```
root
 |-- Pregnancies: integer (nullable = true)
 |-- Glucose: integer (nullable = true)
 |-- BloodPressure: integer (nullable = true)
 |-- SkinThickness: integer (nullable = true)
 |-- Insulin: integer (nullable = true)
 |-- BMI: double (nullable = true)
 |-- DiabetesPedigreeFunction: double (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Outcome: integer (nullable = true)
 |-- features: vector (nullable = true)
```

In [41]:
```
output_data.show(10)
```

```
+-----------+-------+-------------+-------------+-------+----+----
--------------------+---+-------+--------------------+
|Pregnancies|Glucose|BloodPressure|SkinThickness|Insulin| BMI|Diab
etesPedigreeFunction|Age|Outcome|            features|
+-----------+-------+-------------+-------------+-------+----+----
--------------------+---+-------+--------------------+
|          2|    138|           62|           35|     80|33.6|
0.127| 47|      1|[2.0,138.0,62.0,3...|
|          0|     84|           82|           31|    125|38.2|
0.233| 23|      0|[0.0,84.0,82.0,31...|
|          0|    145|           69|           20|     80|44.2|
0.63| 31|      1|[0.0,145.0,69.0,2...|
|          0|    135|           68|           42|    250|42.3|
0.365| 24|      1|[0.0,135.0,68.0,4...|
|          1|    139|           62|           41|    480|40.7|
0.536| 21|      0|[1.0,139.0,62.0,4...|
|          0|    173|           78|           32|    265|46.5|
1.159| 58|      0|[0.0,173.0,78.0,3...|
|          4|     99|           72|           17|     80|25.6|
0.294| 28|      0|[4.0,99.0,72.0,17...|
|          8|    194|           80|           20|     80|26.1|
0.551| 67|      0|[8.0,194.0,80.0,2...|
|          2|     83|           65|           28|     66|36.8|
0.629| 24|      0|[2.0,83.0,65.0,28...|
|          2|     89|           90|           30|     80|33.5|
0.292| 42|      0|[2.0,89.0,90.0,30...|
+-----------+-------+-------------+-------------+-------+----+----
--------------------+---+-------+--------------------+
only showing top 10 rows
```

In [20]:
```
from pyspark.ml.classification import LogisticRegression

final_data = output_data.select(['features','Outcome'])
```

In [21]:
```
final_data.printSchema()
```

```
root
 |-- features: vector (nullable = true)
 |-- Outcome: integer (nullable = true)
```

In [22]:
```
train, test = final_data.randomSplit([0.7,0.3])
models = LogisticRegression(labelCol='Outcome')
model = models.fit(train)
```

In [23]:
```
summary = model.summary
summary.predictions.describe().show()
```

```
+-------+-------------------+-------------------+
|summary|            Outcome|         prediction|
+-------+-------------------+-------------------+
|  count|               1408|               1408|
|   mean|          0.3359375|0.26207386363636365|
| stddev|0.47248497191460576| 0.4399188594808409|
|    min|                0.0|                0.0|
|    max|                1.0|                1.0|
+-------+-------------------+-------------------+
```

In [24]:
```
from pyspark.ml.evaluation import BinaryClassificationEvaluator

predictions = model.evaluate(test)
```

In [25]:
```
predictions.predictions.show(10)
```

```
+--------------------+-------+--------------------+---------------
-----+----------+
|            features|Outcome|       rawPrediction|         probab
ility|prediction|
+--------------------+-------+--------------------+---------------
-----+----------+
|[0.0,67.0,76.0,20...|      0|[2.69918047605536...|[0.936978268386
52...|       0.0|
|[0.0,84.0,64.0,22...|      0|[2.54554450605487...|[0.927273619772
76...|       0.0|
|[0.0,91.0,80.0,20...|      0|[2.55068308495464...|[0.927619391492
40...|       0.0|
|[0.0,93.0,60.0,25...|      0|[2.77921554180796...|[0.941542282507
20...|       0.0|
|[0.0,93.0,100.0,3...|      0|[1.41112603366638...|[0.803943487955
55...|       0.0|
|[0.0,93.0,100.0,3...|      0|[1.41112603366638...|[0.803943487955
55...|       0.0|
|[0.0,94.0,69.0,20...|      0|[2.71473427039555...|[0.937890501296
04...|       0.0|
|[0.0,94.0,70.0,27...|      0|[1.91101198296639...|[0.871132796188
87...|       0.0|
|[0.0,94.0,70.0,27...|      0|[1.91101198296639...|[0.871132796188
87...|       0.0|
|[0.0,95.0,64.0,39...|      0|[1.76307324174124...|[0.853594144357
49...|       0.0|
+--------------------+-------+--------------------+---------------
-----+----------+
only showing top 10 rows
```

In [26]:
```
evaluator = BinaryClassificationEvaluator(rawPredictionCol = 'rawPr
evaluator.evaluate(model.transform(test))
```

Out[26]:
```
0.8383401126991842
```

In [30]:
```
model.save('model1')
```

In [31]:
```
from pyspark.ml.classification import LogisticRegressionModel

model2 = LogisticRegressionModel.load('model1')
```

In [32]:
```
df_test = spark.read.csv('new_test.csv', header=True, inferSchema=T
```

In [33]:
```
df_test.printSchema()
```

```
root
 |-- Pregnancies: integer (nullable = true)
 |-- Glucose: integer (nullable = true)
 |-- BloodPressure: integer (nullable = true)
 |-- SkinThickness: integer (nullable = true)
 |-- Insulin: integer (nullable = true)
 |-- BMI: double (nullable = true)
 |-- DiabetesPedigreeFunction: double (nullable = true)
 |-- Age: integer (nullable = true)
```

In [34]:
```
test_data = assembler.transform(df_test)
```

In [35]:
```
test_data.printSchema()
```

```
root
 |-- Pregnancies: integer (nullable = true)
 |-- Glucose: integer (nullable = true)
 |-- BloodPressure: integer (nullable = true)
 |-- SkinThickness: integer (nullable = true)
 |-- Insulin: integer (nullable = true)
 |-- BMI: double (nullable = true)
 |-- DiabetesPedigreeFunction: double (nullable = true)
 |-- Age: integer (nullable = true)
 |-- features: vector (nullable = true)
```

In [36]:
```
results = model2.transform(test_data)
results.printSchema()
```

```
root
 |-- Pregnancies: integer (nullable = true)
 |-- Glucose: integer (nullable = true)
 |-- BloodPressure: integer (nullable = true)
 |-- SkinThickness: integer (nullable = true)
 |-- Insulin: integer (nullable = true)
 |-- BMI: double (nullable = true)
 |-- DiabetesPedigreeFunction: double (nullable = true)
 |-- Age: integer (nullable = true)
 |-- features: vector (nullable = true)
 |-- rawPrediction: vector (nullable = true)
 |-- probability: vector (nullable = true)
 |-- prediction: double (nullable = false)
```

In [37]:
```
results.select(['features','prediction']).show()
```

```
+--------------------+----------+
|            features|prediction|
+--------------------+----------+
|[1.0,190.0,78.0,3...|       1.0|
|[0.0,80.0,84.0,36...|       0.0|
|[2.0,138.0,82.0,4...|       1.0|
|[1.0,110.0,63.0,4...|       0.0|
+--------------------+----------+
```

In [ ]:

CONCLUSION

This project's major goal was to design and execute Diabetes Prediction Using Machine Learning Methods, which was accomplished successfully. To predict diabetes, the suggested method employs correlation analysis and a feature extraction method, with an accuracy of 83 percent. The experimental results can help health care providers make early predictions and decisions in order to cure diabetes and save people's lives.