DIABETES PREDICTION

It is a fact that nowadays, a lot of people are suffering from diabetes, especially in India as 11.4 percent of the people, almost 101 million people live with diabetes. Accurate and efficient prediction of diabetes in its early stages can help us reduce the amount of people dying with complications. To achieve this practice, we will be using Pyspark MLLIB with logistic regression to predict whether the patient is diabetic or non-diabetic.

Task 1: Install Dependencies & Run Spark Session:
Install the pyspark library, which is necessary for working with Apache Spark.

Create a SparkSession, which is the entry point for interacting with Spark.

TASK 2: Clone & Explore dataset

Get your data: Download the diabetes dataset from the provided link

Read the data: Use spark.read.csv() to turn your data into a DataFrame.

Take a peek: Use df.show() to see the first few rows and df.printSchema() to understand the columns.

```
#clone the diabetes dataset from the github repository
! git clone https://github.com/education454/diabetes_dataset
check if the dataset exists
! ls diabetes_dataset
#create spark dataframe
df = spark.read.csv("/content/diabetes_dataset/diabetes.csv",header = True,
inferSchema = True)
#display the dataframe
df.show()
#print the schema
df.printSchema()
#count the total no. of diabetic and non-diabetic class
print((df.count(), len(df.columns)))
df.groupBy('Outcome').count().show()
#get the summary statistics
df.describe().show()
```

Task 3: Data Cleaning and Preparation
Check for null values and replace them with mean values of that respective column
Remove/replace unnecessary values present in the dataset

```
 #check for null values
for col in df.columns:
```

```
  print(col+":", df[df[col].isNull()].count())
#calculate and replace the unnecessary values by the mean value
from pyspark.sql.functions import *
for i in df.columns[1:6]:
  data = df.agg({i:'mean'}).first()[0]
  print("mean value for {} is {}".format(i,int(data)))
  df = df.withColumn(i,when(df[i]==0,int(data)).otherwise(df[i]))
#find the correlation among the set of input & output variables
for i in df.columns:
  print("Correlation to outcome for {} is {}".format(i,df.stat.corr('Outcome',
i)))
#feature selection
from pyspark.ml.feature import VectorAssembler
assembler =
VectorAssembler(inputCols=['Pregnancies','Glucose','BloodPressure','SkinThickne
ss','Insulin','BMI','DiabetesPedigreeFunction','Age'],outputCol='features')
output_data = assembler.transform(df)
```

Task 4: Correlation Analysis and Feature Selection

Correlation analysis among the input and the output variables.

Selection of the input features `#find the correlation among the set of input & output variables`

```
for i in df.columns:
  print("Correlation to outcome for {} is {}".format(i,df.stat.corr('Outcome',
i)))
feature selection
from pyspark.ml.feature import VectorAssembler
assembler =
VectorAssembler(inputCols=['Pregnancies','Glucose','BloodPressure','SkinThickne
ss','Insulin','BMI','DiabetesPedigreeFunction','Age'],outputCol='features')
output_data = assembler.transform(df)
```

Task 5: Build a Logistic Regression Classifier

Split the Dataset into Training and Testing Set

Build and train the Logistic Regression Model`#create final data`

```
from pyspark.ml.classification import LogisticRegression
final_data = output_data.select('features','Outcome')
#split the dataset(Random split method) ; build the model
train ,test = final_data.randomSplit([0.7,0.3])
models = LogisticRegression(labelCol='Outcome')
```

```
model = models.fit(train)
#summary of the model
summary = model.summary
summary.predictions.describe().show()
```

Task 6: Evaluate & Save the Model
Evaluate and Test the model on the Test Data

Save the Model to the disk

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
predictions = model.evaluate(test)
evaluator =
BinaryClassificationEvaluator(rawPredictionCol='rawPrediction',labelCol='Outcom
e')
evaluator.evaluate(model.transform(test))
```

Load the saved Logistic Regression Model on Colab to further predict on the new set of data

Task 7: Model Prediction on a New Set of Unlabelled Data
Create a Spark Dataframe

Use the Saved Model to Predict Diabetes on the New Set of Test Data

```
#create a new spark
dataframe
test_df =
spark.read.csv("/content/diabetes_dataset/new_test.csv",header=True,inferSchema
=True)
#use model to make predictions
results = model.transform(test_data)
results.printSchema()
```

RESULTS

```
#display the predictions
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...|       1.0|
+--------------------+----------+
```