DIABETES RISK PREDICTION

CPSC 531 Advanced Database Management

Team Members: Anusha Anandhan (CWID - 885176289)

Pravallika Bahadur (CWID - 885177543)

INTRODUCTION:

Diabetes is a chronic disease that affects millions of people worldwide, and it can lead to severe health complications if left untreated. Early diagnosis and timely management of diabetes can significantly reduce the risk of complications, making it crucial to identify people at risk of developing the disease. With the increasing availability of big data and machine learning tools, there is a huge opportunity to leverage these technologies to develop accurate and efficient models for diabetes risk prediction. The main goal of this project is to predict the risk of diabetes in the patient in accordance with medical and lifestyle history by using distributed job processing with the help of Google Cloud Platform and PySpark.

PROJECT SCOPE:

- The main scope of the project is to predict the risk of diabetes in the patient.
- The prediction is done by analyzing the health and lifestyle factors of the patient based on lifestyle changes and previous health history.
- We will predict whether the patient has the risk of getting diabetes or whether he/she is in the stage of getting prediabetes or he/she is affected by diabetes.

TECHNOLOGIES USED:

- OS: Windows
- Framework: Google Cloud Platform (GCP), Spark, BiqQuery, Streamlit
- Languages Used: Python
- Libraries: Pyspark, Matplotlib
- IDE: Jupyter Notebook

FUNCTIONALITIES:

1. **DATASET:**

   The Diabetes data set has been collected from Kaggle https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset?resource=download .

   This Dataset has 13253680 rows and 22 columns which have features like 'Diabetes_012', 'HighBP', 'HighChol', 'CholCheck', 'BMI', 'Smoker', 'Stroke', 'HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies', 'HvyAlcoholConsump', 'GenHlth', 'MentHlth', 'PhysHlth', 'DiffWalk', 'Sex', 'Age'

2. **Data Cleaning:**

- Data cleaning is the act of locating and fixing mistakes, inconsistencies, and inaccuracies in data. To improve the reliability and validity of the analysis, data cleaning is needed in the data analysis process. It makes sure that the data is accurate, full, and consistent.
- Typically, data cleansing requires a number of procedures, including:

- Eliminating duplicate records from the dataset entails finding and eliminating any such records.
- Managing missing data: Missing data is a prevalent problem in datasets and can be managed by either eliminating records with missing values or imputing missing values using statistical methods.

The diabetes dataset has some features that are inconsistent, so we dropped the features like Income, Education, NoDocbcCost, AnyHealthcare.

This dataset doesn't contain missing/null values.

```
#null values sum
data.isnull().sum()
```

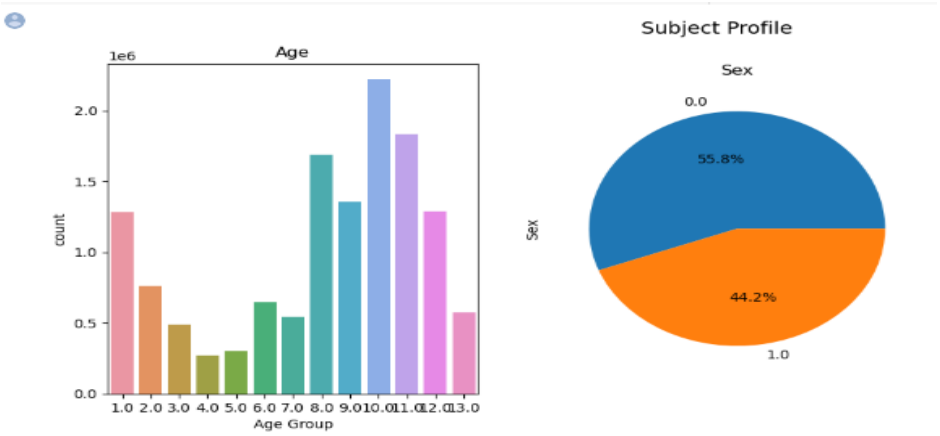
```
Diabetes_012            0
HighBP                  0
HighChol                0
CholCheck               0
BMI                     0
Smoker                  0
Stroke                  0
HeartDiseaseorAttack    0
PhysActivity            0
Fruits                  0
Veggies                 0
HvyAlcoholConsump       0
GenHlth                 0
MentHlth                0
PhysHlth                0
DiffWalk                0
Sex                     0
Age                     0
dtype: int64
```

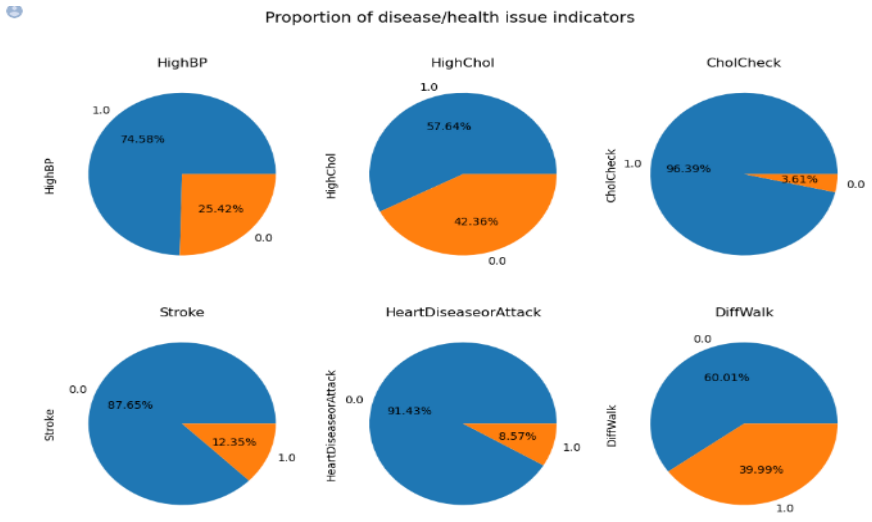
3. Data Analysis:

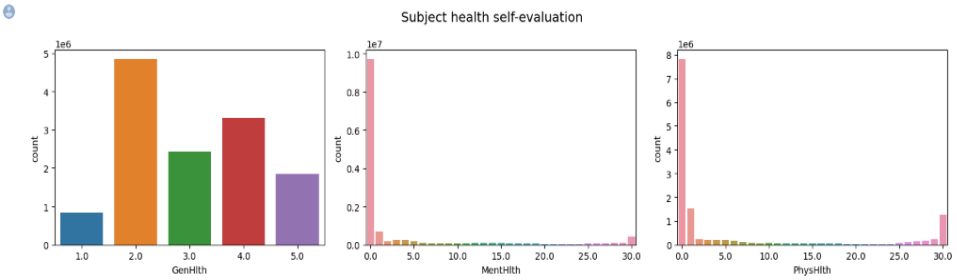
- PySpark is a distributed computing framework that offers a strong foundation for handling massive amounts of data. By enabling data to be processed in parallel across numerous nodes in a cluster, Spark is made to handle big data tasks. Spark is a popular option for large data analytics since it also offers a wealth of data analysis

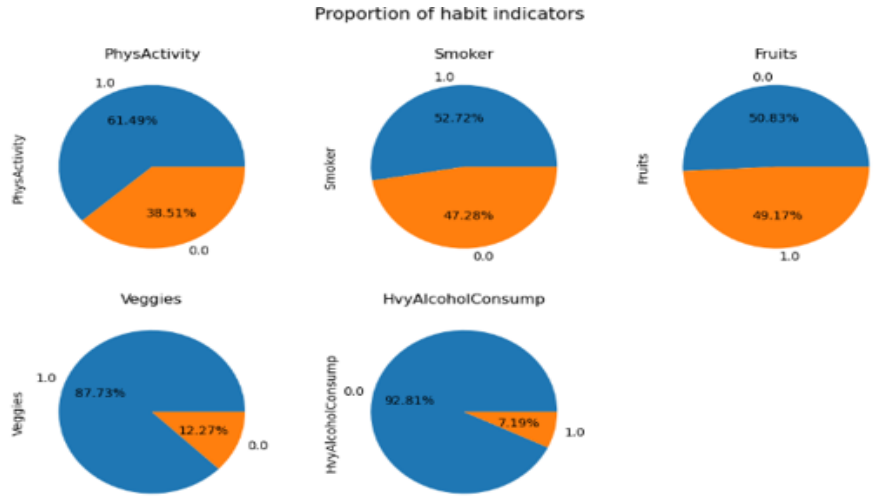
APIs and libraries. Here we have analyzed the data by using the MatplotLib where the different features of the dataset have been plotted so that it will create a better picture of features and the correlations between the features which in turn will help for the feature selection.

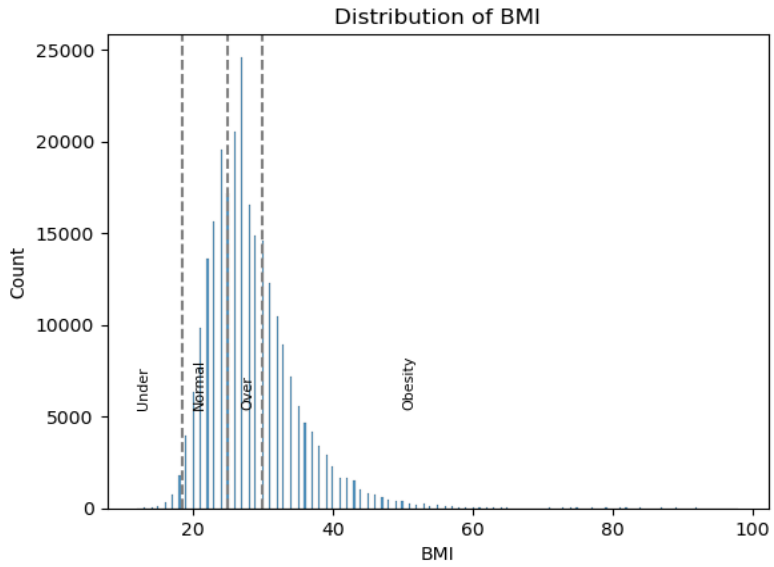
The goal of feature selection techniques in machine learning is to find the best set of features that allows one to build optimized models. For this project, we are using Diabetes_012, HighBP, HighChol, CholCheck, BMI, Smoker, Stroke, HeartDiseaseorAttack, PhysActivity, HvyAlcoholConsump, GenHlth, MentHlth, PhysHlth, DiffWalk, Sex, Age as the required features





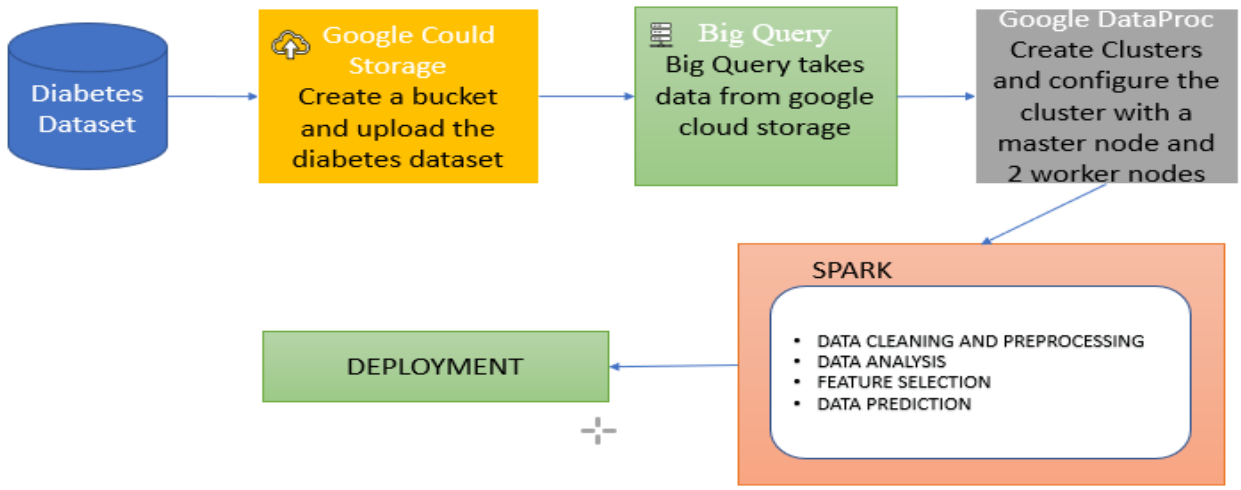

Subject health self-evaluation
1e6
count
GenHlth
1e7
MentHlth
1e6
PhysHlth


Proportion of habit indicators
PhysActivity
61.49%
38.51%
Smoker
52.72%
47.28%
Fruits
50.83%
49.17%
Veggies
87.73%
12.27%
HvyAlcoholConsump
92.81%
7.19%


Distribution of BMI
Count
BMI
Under
Normal
Over
Obesity

4. **BiqQuery**: BigQuery is a popular data warehouse service that allows you to easily work with large amounts of data. By separating the computational engine that analyzes data from your storage options, BigQuery enhances flexibility. BigQuery can be used to evaluate the data's location or to store and analyze your data there. While streaming enables continuous data updates, federated queries allow you to read data from external sources. You can analyze and comprehend that data with the help of potent tools like BigQuery ML and BI Engine.
5. **Google Cloud Platform**: This is a cloud platform where we can develop our distributed system and submit the distributed job that has been implemented. The data has been managed by google cloud storage where we need to create a bucket and upload the dataset that has been for the prediction. This can be done from the Google Cloud Storage console. Once the data has been stored successfully, we can start creating the cluster by using Dataproc. Here we will configure our cluster by having one master node and two worker nodes, and then run the jobs on the Spark cluster and perform prediction using the Jupiter Notebook where we will use PySpark libraries.

ARCHITECTURE AND DESIGN:



1) Google Cloud Storage: Google cloud storage is one of the functionalities of Google cloud platform (GCP) where we can upload our dataset files. Here for our diabetes risk prediction, we have uploaded the diabetes dataset to the Google cloud storage that is available in the GCP console. Then the dataset is loaded by creating the bucket i.e. diabetesbucket.

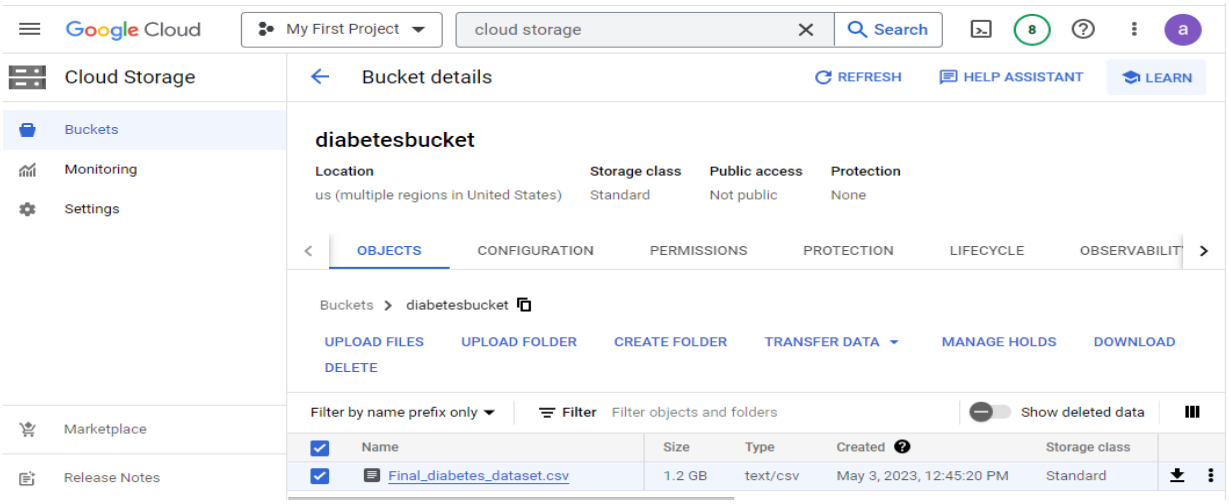


Figure 1: Loading the dataset in Google Could Storage

2) Bigquery is used to take and read the data from cloud storage where it will convert the data into a relational database from where the data can be accessed and analysis and prediction can be performed. This is done by creating a dataset and a table under it by using Create dataset and Create table option. Once the data has been linked from the cloud storage to Bigquery, we could preview the table successfully.

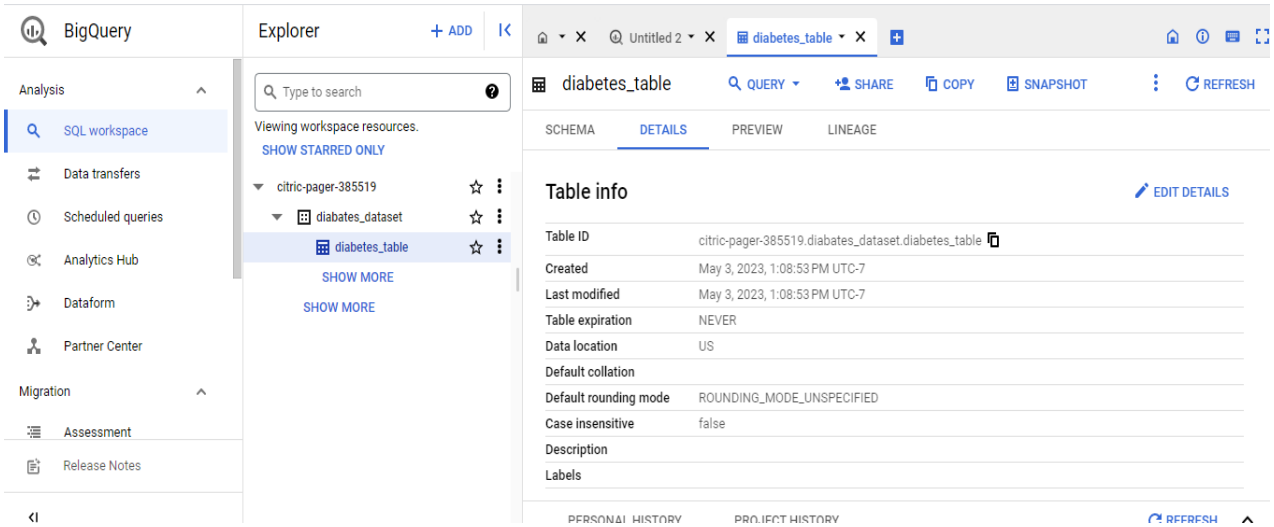


Figure 2: Table creation in Bigquery

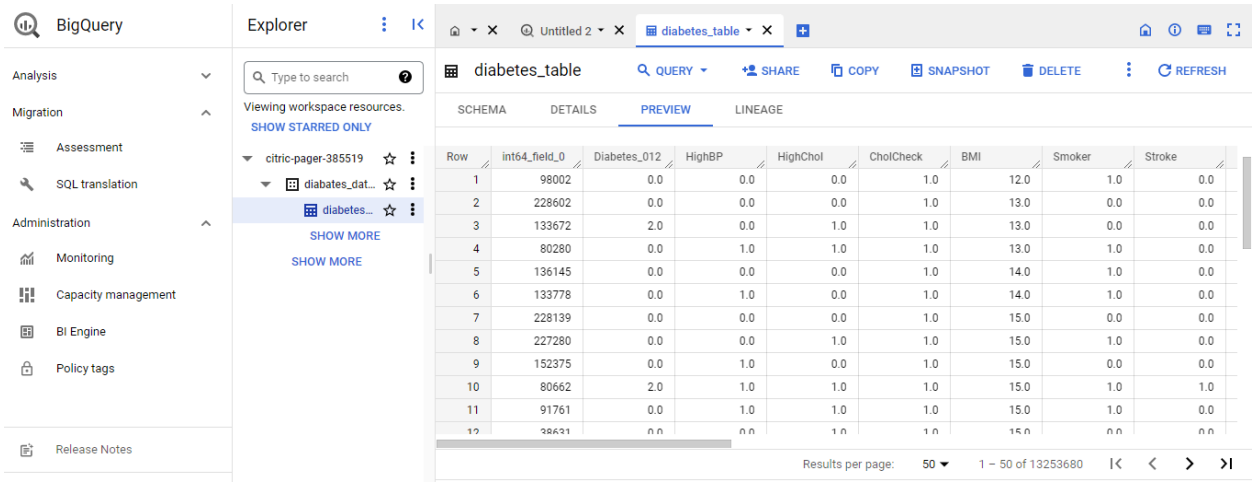


Figure 3: Table Preview in Bigquery

3) Google DataProc: Next step is to create a Spark job by creating clusters in Dataproc where we need to use the create cluster option from the Dataproc console and start configuring it. For this project, we have used one master node and two worker nodes and we are also enabling the gateway to use the Jupyter Notebook where we could do our implementation once the Spark session has been created.

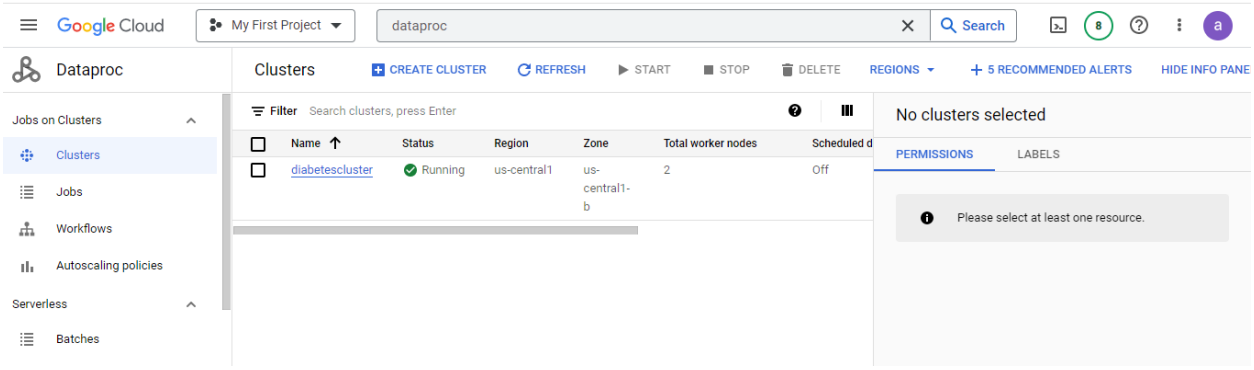


Figure 4: Cluster creation in DataProc

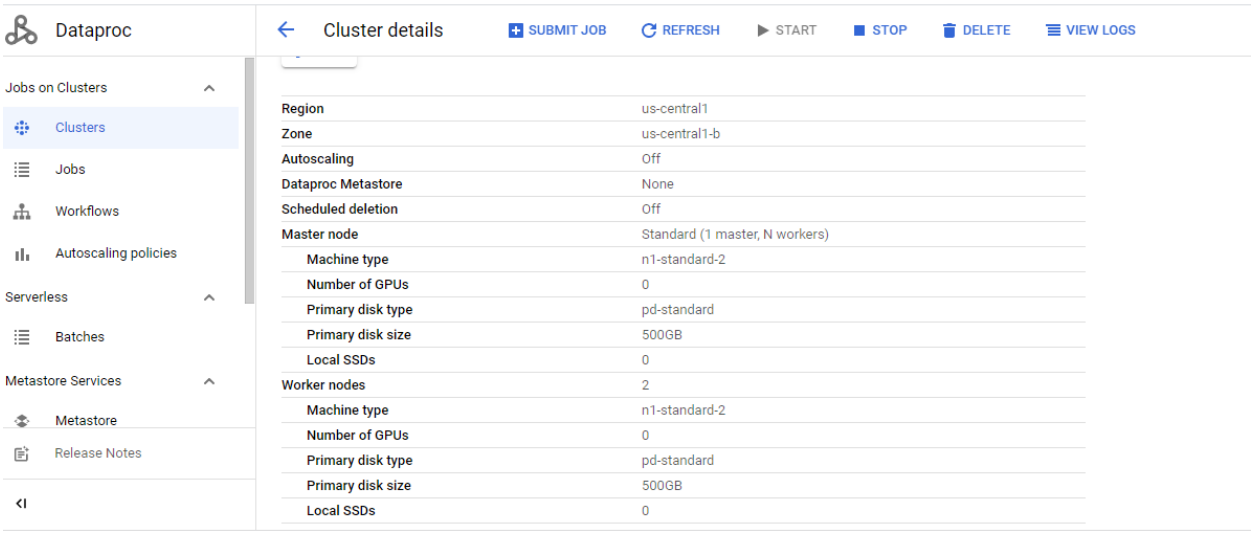


Figure 5: Diabetes Cluster Configuration

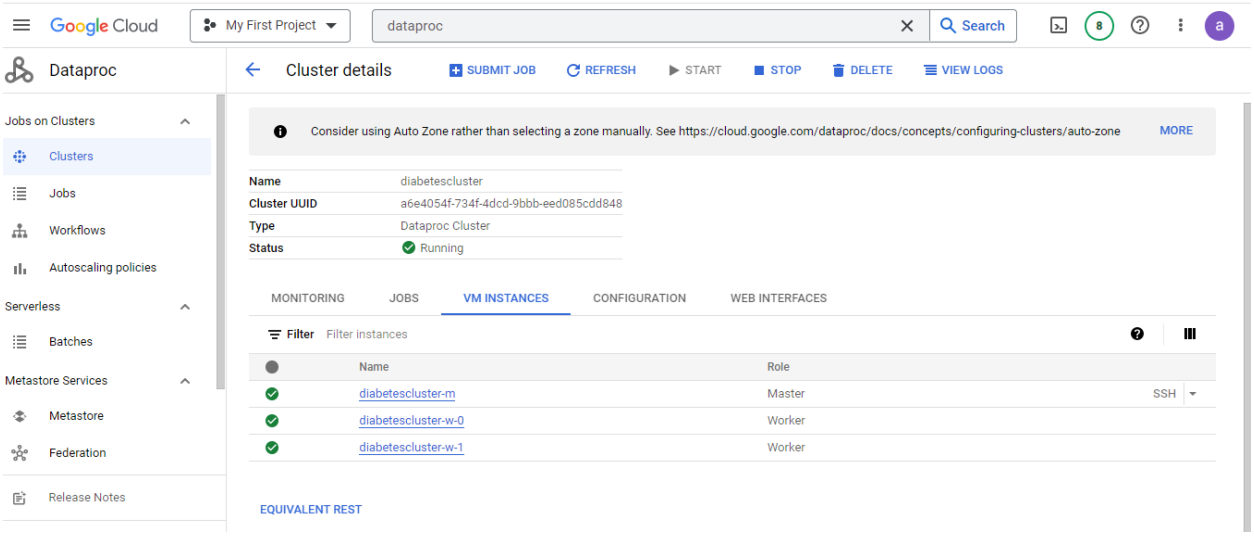


Figure 6: Master and Worker node running instances in Dataproc

4) Spark: After creating the cluster, once the Spark session has been started, we are implementing the data cleaning, analysis, feature selection, and prediction by using PySpark using Jupyter Notebook. Once data cleaning and analysis have been done, we are doing the feature selection which is done by analyzing the heat map which helps to understand the correlation between the features.

Figure 7: HeatMap

Once feature prediction is done, we are using the Random Forest classification machine-learning algorithm. Here we are splitting up 20 % of the data as a test dataset and 80% as a training dataset. By using this model, an accuracy of 97% has been achieved.

```
[11] (training_data, test_data) = transformed_data.randomSplit([0.8,0.2])

[13] from pyspark.ml.classification import RandomForestClassifier

     lr = RandomForestClassifier(featuresCol = 'features', labelCol = 'Diabetes_012')
     lrModel = lr.fit(training_data)

[14] rf_predictions = lrModel.transform(test_data)

[15] from pyspark.ml.evaluation import MulticlassClassificationEvaluator

     multi_evaluator = MulticlassClassificationEvaluator(labelCol = 'Diabetes_012', metricName = 'accuracy')
     print('Random Forest classifier Accuracy:', multi_evaluator.evaluate(rf_predictions))

Random Forest classifier Accuracy: 0.9713666840444257
```

DEPLOYMENT:

We have created a User Interface (UI) using Streamlit for deploying the prediction model that has been running in the GCP cluster.

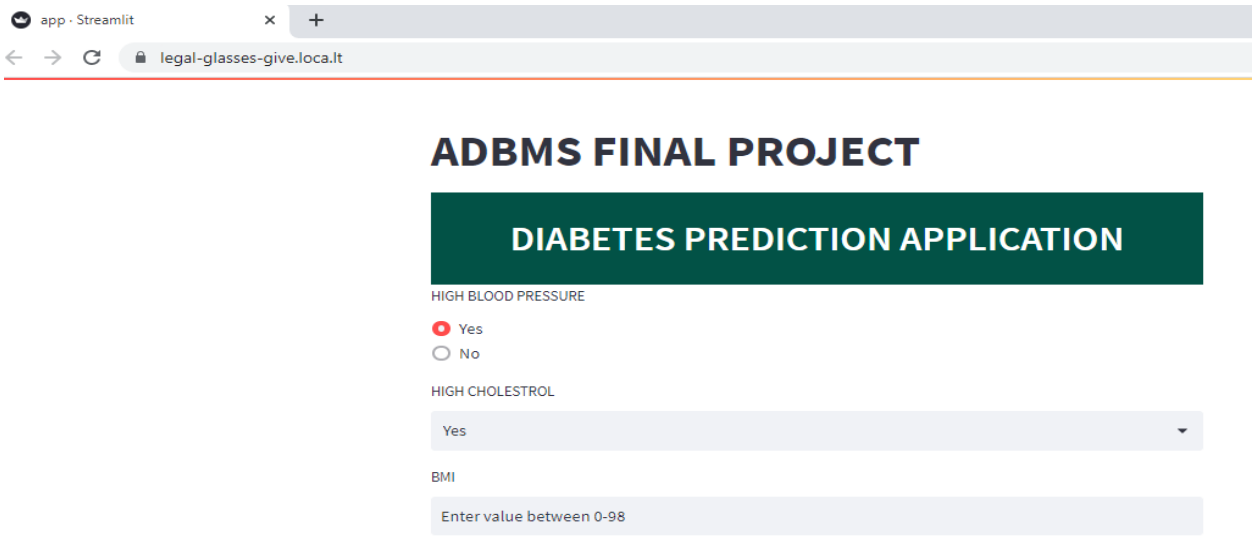


Figure 8: Diabetes Prediction Application using Streamlit

Github Code:

Analysis Code:
https://github.com/anusha562/Diabetes_prediction/blob/main/Analysis_Diabetes.ipynb

Prediction Code:
https://github.com/anusha562/Diabetes_prediction/blob/main/Diabetes_Prediction_FinalAdbms.ipynb

Deployment Code:

https://github.com/anusha562/Diabetes_prediction/blob/main/diabetes_pred_UI.py

DEPLOYMENT INSTRUCTIONS:

1) Download the trained model from the cluster.

2) Use the model for prediction through Streamlit App (Diabetes Prediction Application)
3) Run the application by using “streamlit run application_name” command
4) Once the app is up, user inputs can be given from the features provided.
5) Predict diabetes by hitting the predict button

STEPS TO RUN:

Step 1: Login to the Google Cloud Platform (GCP) application

Step 2: Load the dataset into the Google Cloud Storage

Step 3: Link the dataset from the Google Cloud Storage to BigQuery

Step 4: Create cluster in the Dataproc to start the Spark session

Step 5: Once cluster is up, do the analysis for all the feature set.

Step 6: Perform prediction using the PySpark using Jupyter Notebook as the IDE

Step 7: Create a web application using Streamlit for diabetes application

Step 8: Run the web app and do prediction by using the model implemented in the cluster.

TEST RESULTS:

1. Diabetes prediction has been achieved with 97% accuracy using Big data technology and machine learning.

```
[11] (training_data, test_data) = transformed_data.randomSplit([0.8,0.2])
```

```
[13] from pyspark.ml.classification import RandomForestClassifier

     lr = RandomForestClassifier(featuresCol = 'features', labelCol = 'Diabetes_012')
     lrModel = lr.fit(training_data)
```

```
[14] rf_predictions = lrModel.transform(test_data)
```

```
[15] from pyspark.ml.evaluation import MulticlassClassificationEvaluator

     multi_evaluator = MulticlassClassificationEvaluator(labelCol = 'Diabetes_012', metricName = 'accuracy')
     print('Random Forest classifier Accuracy:', multi_evaluator.evaluate(rf_predictions))
```

```
Random Forest classifier Accuracy: 0.9713666840444257
```

Confusion Matrix:

```
#plot the confusion matrix
df = rf_predictions
tp = df[(df.Diabetes_012 == 1) & (df.prediction == 1)].count()
tn = df[(df.Diabetes_012 == 0) & (df.prediction == 0)].count()
fp = df[(df.Diabetes_012 == 0) & (df.prediction == 1)].count()
fn = df[(df.Diabetes_012 == 1) & (df.prediction == 0)].count()
print ("True Positives:", tp)
print ("True Negatives:", tn)
print ("False Positives:", fp)
print ("False Negatives:", fn)
print ("Total", df.count())

r = float(tp)/(tp + fn)
print ("recall", r)

p = float(tp) / (tp + fp)
print ("precision",p)
```

```
True Positives: 1409208
True Negatives: 441605
False Positives: 75904
False Negatives: 0
Total 2650898
recall 1.0
precision 0.9488900500433637
```

```
[ ] confusion matrix
    [1409208 0
     75904   441605]
```