

CS5811 DISTRIBUTED DATA ANALYSIS

STUDENT NAME : NAZEEMA BEGUM RAHMAN BASHSA

STUDENT ID : 2351211

TABLE OF CONTENT

1. DATA DESCRIPTION AND RESEACH QUESTION	2
2. DATA PREPARATION AND CLEANING	2
3. EXPLORATORY DATA ANALYSIS	5
4. MACHINE LEARNING PREDICTION	7
5.HIGH PERFORMANCE COMPUTATIONAL IMPLEMENTATION.	8
6. PERFORMANCE EVALUATION AND COMPARISON OF METHODS	9
7.DISCUSSION OF THE FINDINGS	10
8. DATA MANAGEMENT PLAN AND AUTHOR CONTRIBUTION STATEMENT.	10
9. APPENDIX	11

DATA DESCRIPTION:

The dataset used in this project is a diabetes prediction dataset. The dataset is retrieved from the Kaggle ¹. This dataset includes patients' medical and demographic information, as well as their diabetes status (positive or negative). It include information such as age, gender, Body Mass Index (BMI), hypertension, heart disease, smoking history, HbA1c level, and blood glucose level.

Research Question:

“What is the likelihood to develop diabetes in a patient based on their medical and demographic details given?”

DATA PREPARATION AND CLEANING

The dataset consists of 50k rows which is retrieved in the form of .csv file. The process begins with loading the dataset, usually in the .csv format, using RStudio. The initial steps is to installation of the packages required for the process. The packages includes stats, dplyr, ggplot2, ggfortify, cluster, randomforest, and caret and the libraries are to be loaded. Read.csv function is used to read the csv file, with the na.string option set to “NA” and an empty string “”. This parameter controls how missing values are represented in the dataset.

The next step is looking at the dataset, which involves summary function. The summary function provides statistics for numeric vectors include minimum, maximum, mean, median, and quartiles. The apply(is.na()) checks the missing values (NA). The output shows the percentage of missing values for each column in the dataset.

```
#### check to see the count of missing values in the categorical columns smoking history and diabetes
```{r}
table(diabetes_dataset$smoking_history)
table(diabetes_dataset$diabetes)

analysis of the continuous variables bmi, HbA1c_level and blood_glucose_level
colSums(is.na(diabetes_dataset))

missing_bmi <- is.na(diabetes_dataset$bmi)
median_bmi <- median(diabetes_dataset$bmi, na.rm = TRUE)
diabetes_dataset$bmi[missing_bmi] <- median_bmi

missing_HbA1c_level <- is.na(diabetes_dataset$HbA1c_level)
median_HbA1c_level <- median(diabetes_dataset$HbA1c_level, na.rm = TRUE)
diabetes_dataset$HbA1c_level[missing_HbA1c_level] <- median_HbA1c_level

missing_blood_glucose_level <- is.na(diabetes_dataset$blood_glucose_level)
median_blood_glucose_level <- median(diabetes_dataset$blood_glucose_level, na.rm = TRUE)
diabetes_dataset$blood_glucose_level[missing_blood_glucose_level] <-
median_blood_glucose_level
```
```

Fig 1

Imputing the missing values to get a cleaned dataset shown fig 3. It is critical to understand the distribution of missing values in categorical columns such as smoking history and diabetes status. To know the completeness of missing values in the columns using R's table() function. The output after imputation is shown below fig 4.

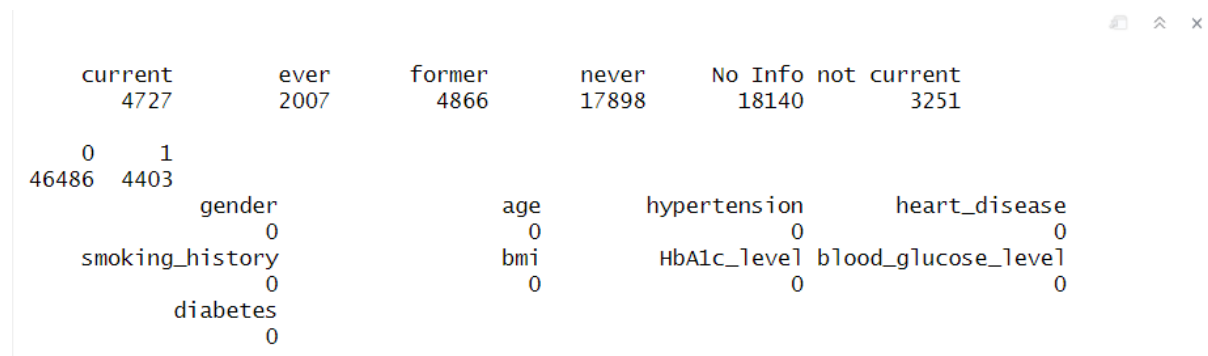


Fig 2

Outliers should be handled after imputing the missing values. Using box plot checking whether any outlier present in the dataset.

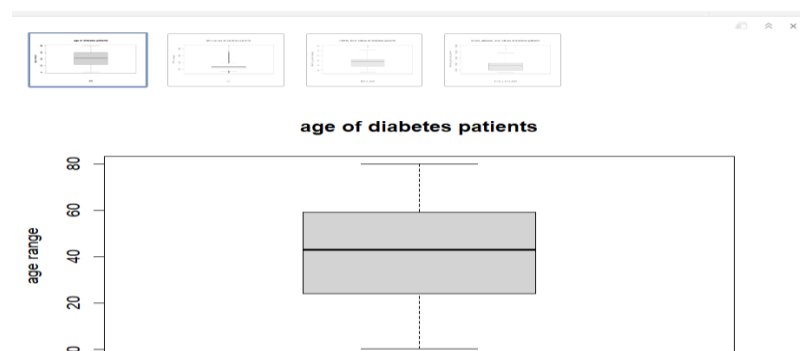


Fig 3

Boxplot analysis reveals that the age column lacks outliers, however the BMI column has a significant number of outliers. Because of the inherent fluctuations in BMI measurements caused by an individual's height and weight, eliminating outliers from this dataset is tough.

Variations in HbA1c and blood_glucose_level, which are important markers for diabetic patients, can occur owing to a variety of factors such as eating patterns, physical activity, medication, and stress. As a result, the choice is made not to remove outliers in these variables, acknowledging their potential importance within diabetes populations. There are no outliers, identified for the binary variables, hypertension and heart_disease, which are limited to values 0 and 1. Similarly, because the gender column is categorical, it naturally lacks outliers.

EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA) is a critical phase in data analysis, requiring the first investigation and comprehension of a dataset before applying more advanced analyses. K-mean cluster analysis is used for EDA part. K-means clustering is an unsupervised machine learning algorithm that divides a dataset into a predetermined number of clusters (K clusters) based on similarities. The algorithm assigns data points repeatedly to the nearest cluster centroid and then recalculates the centroids until convergence is achieved. It seeks to minimize variance inside clusters while maximizing variance between clusters. K-means is widely utilized in a variety of applications, including data mining, pattern recognition, and image segmentation.

K-mean clustering is best suited for diabetic prediction because it can detect discrete groups within heterogeneous diabetic populations. By determining key clinical factors such as age, gender, BMI, and blood pressure, K-mean can effectively divide patients into groups based on shared characteristics. This clustering helps in the identification of common diabetes patterns or subtypes, allowing for more individualized treatment plans and interventions. Furthermore, K-mean clustering can handle enormous datasets rapidly, making it applicable for studying varied patient cohorts and generating useful insights for diabetic predictions.

Clustering can be utilized in diabetic prediction by identifying patterns within patients' dataset. Clustering begins with converting the categorical variables like gender, and smoking history into numeric format for compatibility with clustering algorithms. This allows us to enable all relevant data in the analysis.

The 'wssplot' function computes the within-group sum of squares for various cluster configurations. This helps to calculate the ideal number of clusters. The graphic depicts the elbow point. The prepared dataset is clustered using k-means, with the exception of the target variable "diabetes" column. 'cluster ID' supplied to each point denotes its participation in a certain cluster. This assignment is based on how closely the data points match the cluster centroids. The k_cluster_id_diabetes variables save these cluster IDs for subsequent investigation or interpretation.

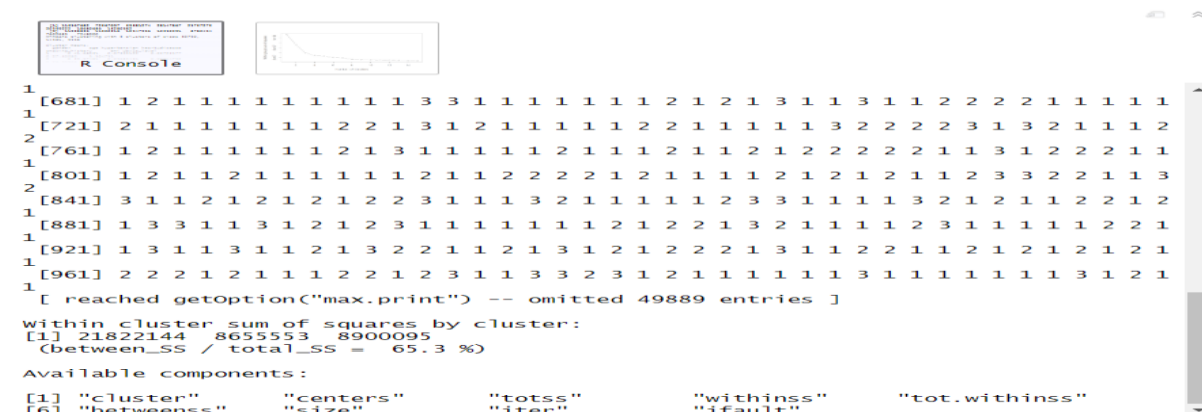


Fig:4

The output on the fig shows the components assigned during the cluster analysis execution.

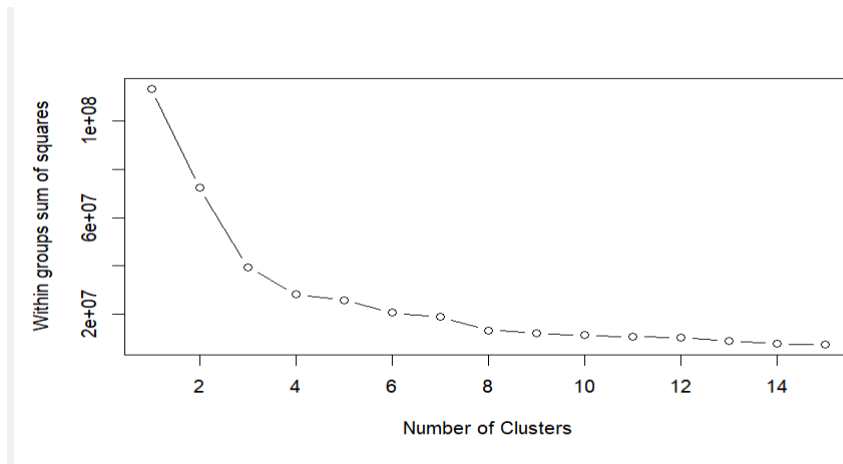


Fig:5

In this graph, the x-axis represents the number of clusters, and y-axis represent the within-group sum of squares. The plot shows how the 'wss' changes as the number of clusters increases. A cluster plot plays a vital role for visualising a cluster analysis, such as k-means.

The cluster results generated from the k-means clustering algorithm by determining the average value of attributes inside each cluster.

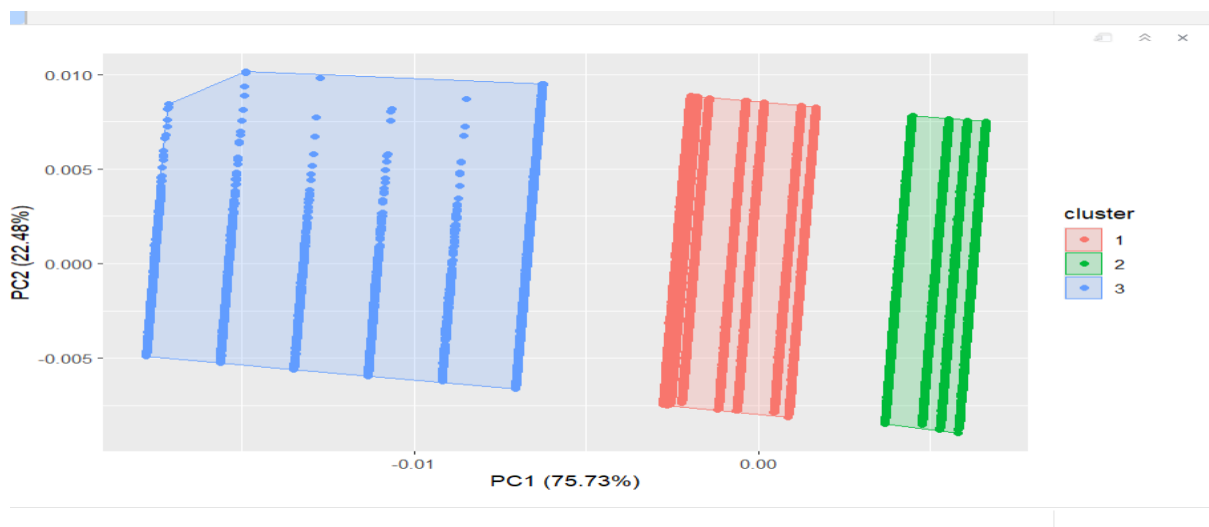


Fig : 6

This researches highlights the unique characteristics of each cluster's centroids, demonstrating significant patterns in the data. Cluster 1 has an average age of 41.68 years, with relatively low rates of hypertension and cardiac arrests, as well as moderate BMI, HbA1c, and blood glucose level.

Cluster 2 has a significantly younger group, with an average age of 39.97 years and lower BMI, HbA1c, and blood glucose levels. However, there is a somewhat higher frequency of hypertension and heart disease than in cluster 1.

Cluster 3 stands out for its average age of 47.8 and considerably higher levels of blood glucose, HbA1c, and BMI. Furthermore, a sizable proportion of people in this cluster have hypertension and cardiac disease.

Machine Learning Prediction

The Machine learning prediction techniques used for the diabetic dataset is random forest Technique. Random forest is a machine learning technique that employs the ensemble learning principle by training many decision trees and generating the mode of the classes (classification) or the mean prediction (Regression) of individual trees. Each tree in the forest is trained using a random subset of training data and characteristics. The randomisation reduces overfitting and improves the models generalisation performance. Introducing seed value of 123 initializes the random number to ensure reproducibility of results. By using the training_index, 70% of rows are randomly selected from the dataset.

Constructs the Random Forest model. It predicts the 'diabetes' outcome variable based on all other variables in the training dataset. The model generates 500 decision trees and selects the square root of the total number of variables at each split.

By publishing a summary of the trained random forest model that contains information such as the number of trees, the mean reduction in accuracy, and other relevant metrics.

The output of the random forest classifier interprets the results of the performance matrices. The model shows an OOB estimate of error rate of 2.93%. The OOB error provides an actual measure of its performance.

It aids in testing the Random Forest model's generalization potential without requiring a separate test set. The confusion matrix generates shows the True negative value of 32468, and false positive value of 33, false negative = 1009 and true positive = 2112.

The error rate in the negative class (non-diabetic) is about 0.001%. The error rate in the positive class (diabetes) is around 0.32%.

```
Call:
  randomForest(formula = diabetes ~ ., data = training_diabetes,      ntree = 500, mtry =
    sqrt(7))

      Type of random forest: classification
      Number of trees: 500
No. of variables tried at each split: 3

      OOB estimate of  error rate: 2.93%
Confusion matrix:
      0      1 class.error
0 32468    33 0.001015353
1  1009 2112 0.323293816
```

Fig 7

By plotting the rf_diabetes it provide a error rate curve shown below Fig 8

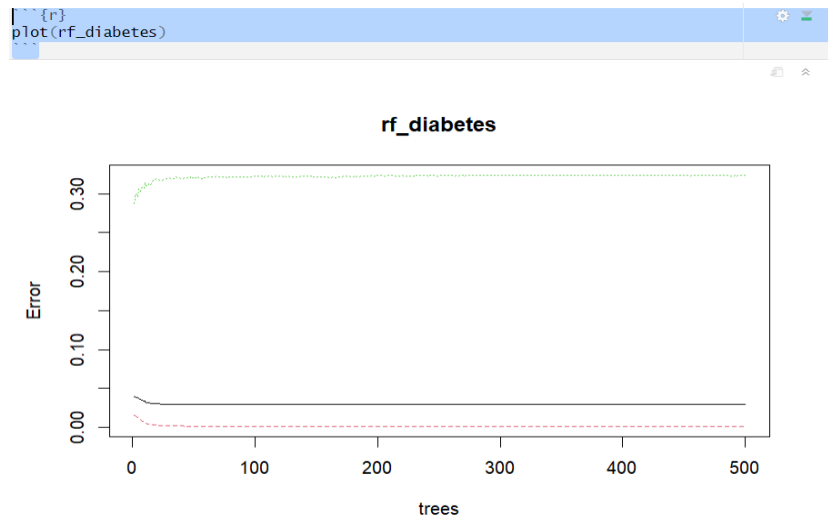


Fig 8

By evaluating the performance model, the accuracy shows 0.9718 fig 9

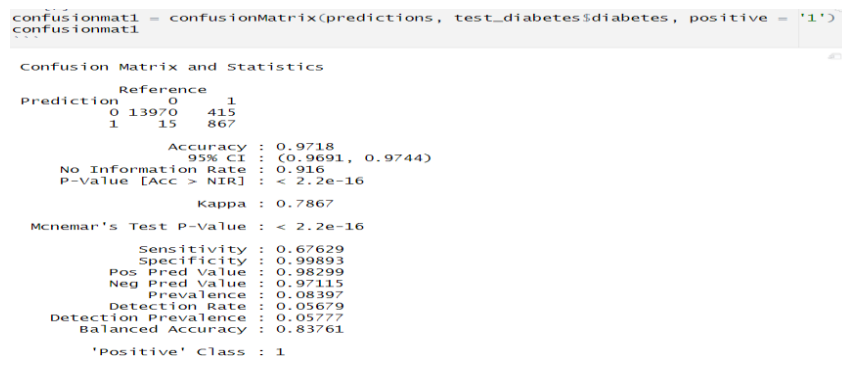


Fig 9

5. High Performance Computational implementation

Pyspark is the HPC technique which is used in this project. Random Forest in Spark uses distributed computing to attain great performance. Spark distributes data across numerous nodes in a cluster, enabling simultaneous processing of the data. Each node works independently on a part of the data.

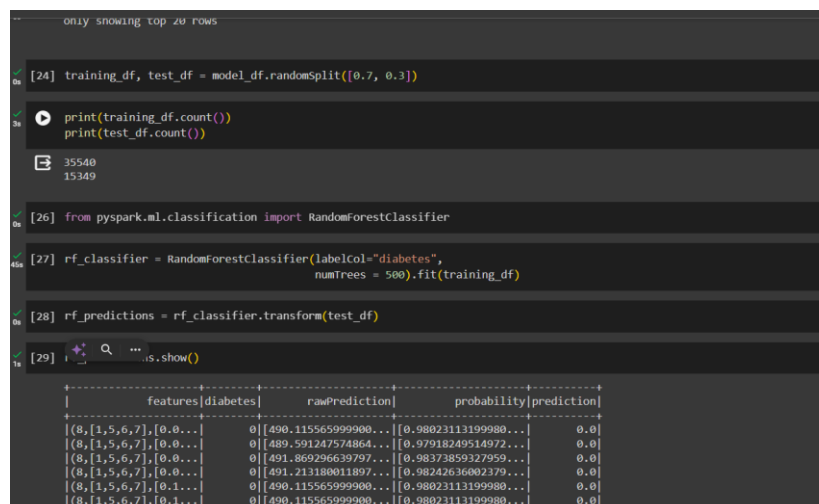
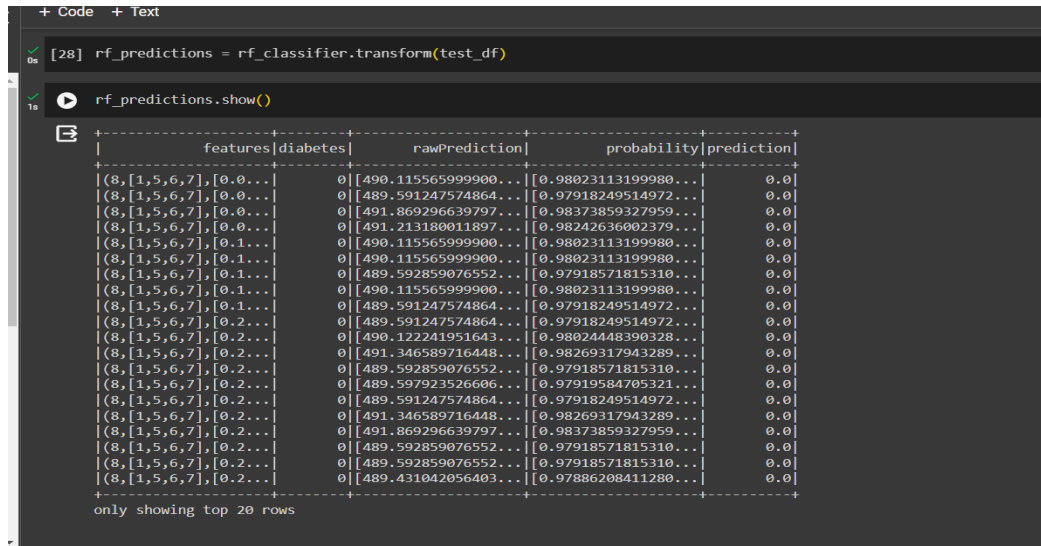


Fig 10

Random Forest builds each tree independently, allowing numerous trees to be constructed in parallel. Spark efficiently distributes tree construction duties across worker nodes, accelerating the process. By Installing the pyspark on the virtual machine in the google colab, random forest is carried out using “!pip install pyspark”. With python code on pyspark create a train and test dataset (0.7.0.3) and get the results. Rf_prediction shows the prediction output as mentioned below.



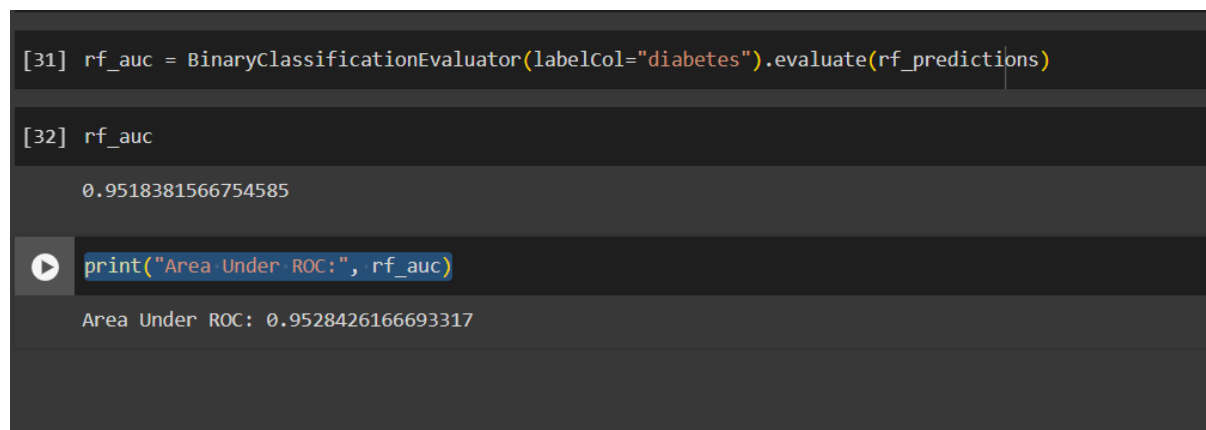
```
[28] rf_predictions = rf_classifier.transform(test_df)
rf_predictions.show()
```

| features | diabetes | rawPrediction | probability | prediction |
|------------------------|----------|----------------------|----------------------|------------|
| (8,[1,5,6,7],[0.0...]) | 0 | [490.115565999900... | [0.98023113199980... | 0.0 |
| (8,[1,5,6,7],[0.0...]) | 0 | [489.591247574864... | [0.97918249514972... | 0.0 |
| (8,[1,5,6,7],[0.0...]) | 0 | [491.869296639797... | [0.98373859327959... | 0.0 |
| (8,[1,5,6,7],[0.0...]) | 0 | [491.213180011897... | [0.98242636002379... | 0.0 |
| (8,[1,5,6,7],[0.1...]) | 0 | [490.115565999900... | [0.98023113199980... | 0.0 |
| (8,[1,5,6,7],[0.1...]) | 0 | [490.115565999900... | [0.98023113199980... | 0.0 |
| (8,[1,5,6,7],[0.1...]) | 0 | [489.592859076552... | [0.97918571815310... | 0.0 |
| (8,[1,5,6,7],[0.1...]) | 0 | [490.115565999900... | [0.98023113199980... | 0.0 |
| (8,[1,5,6,7],[0.1...]) | 0 | [489.591247574864... | [0.97918249514972... | 0.0 |
| (8,[1,5,6,7],[0.2...]) | 0 | [489.591247574864... | [0.97918249514972... | 0.0 |
| (8,[1,5,6,7],[0.2...]) | 0 | [490.122241951643... | [0.98024448390328... | 0.0 |
| (8,[1,5,6,7],[0.2...]) | 0 | [491.346589716448... | [0.98269317943289... | 0.0 |
| (8,[1,5,6,7],[0.2...]) | 0 | [489.592859076552... | [0.97918571815310... | 0.0 |
| (8,[1,5,6,7],[0.2...]) | 0 | [489.597923526606... | [0.97919584705321... | 0.0 |
| (8,[1,5,6,7],[0.2...]) | 0 | [489.591247574864... | [0.97918249514972... | 0.0 |
| (8,[1,5,6,7],[0.2...]) | 0 | [491.346589716448... | [0.98269317943289... | 0.0 |
| (8,[1,5,6,7],[0.2...]) | 0 | [491.869296639797... | [0.98373859327959... | 0.0 |
| (8,[1,5,6,7],[0.2...]) | 0 | [489.592859076552... | [0.97918571815310... | 0.0 |
| (8,[1,5,6,7],[0.2...]) | 0 | [489.592859076552... | [0.97918571815310... | 0.0 |
| (8,[1,5,6,7],[0.2...]) | 0 | [489.431042056403... | [0.97886208411280... | 0.0 |

only showing top 20 rows

Fig 11

The evaluator used to assess the performance of binary classification models. It computes metrics such as the area under the ROC curve (Receiver Operating Characteristic curve), which is a typical statistic used to evaluate binary classification algorithms. Fig 12 shows the accuracy of the random forest classifier as 0.952



```
[31] rf_auc = BinaryClassificationEvaluator(labelCol="diabetes").evaluate(rf_predictions)
[32] rf_auc
0.9518381566754585
print("Area Under ROC:", rf_auc)
Area Under ROC: 0.9528426166693317
```

Fig 12

PERFORMANCE EVALUATION AND COMPARISON OF METHODS

In this instance, R achieves a greater accuracy of 0.97 than Spark (0.95). This suggests that the model developed in R is marginally more accurate in its predictions.

Both R and Spark may have different algorithm implementations, resulting in varying model performance. Decision trees are simple and interpretable, making them easy to grasp and depict. However, they may struggle with complicated datasets and are susceptible to overfitting. Random forests, on the other hand, are an ensemble of decision trees that generalize better and are less prone to

overfitting. They often improve accuracy by combining predictions from numerous trees. While decision trees give explicit decision-making criteria, random forests improve robustness and performance, particularly for complicated datasets found in medical applications such as diabetes prediction.

DISCUSSION AND FINDING

While implementing the decision tree in R the accuracy of 0.97 percent is attained and random forest the accuracy is 0.97. whereas the random forest in spark produce the accuracy of 0.95 which is less than random forest.

Data Management Plan and Author Contribution statement

- **Data Management Plan:**

1. Overview

| | |
|-------------------|----------------------------|
| Researcher: | Nazeema begum Rahman basha |
| Project title: | Diabetes Prediction |
| Project duration: | 150 hours |
| Project context: | predict diabetes |

2. Defining your data/research sources

| |
|---|
| 2.1 Where will your data/research sources come from?
https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset |
| 2.2 How often will you get new data?
Often as it is patients records |
| 2.3 How much data/information will you generate?
Double the data in 2 months |
| 2.4 What file formats will you use?
csv file format. |

3. Organising your data

| |
|---|
| 3.1 How will you structure and name your folders and files?
2351211.pdf |
| 3.2 What additional information is required to understand each data file?
Its just the pdf report, rmd file and python file |
| 3.3 What different versions of each data file or source will your create?
There is only one data file that has been used which is diabetes_prediction.csv file |

4. Looking after your data

| |
|---|
| 4.1 Where will you store your data?
it is saved on drive. |
| 4.2 How will your data be backed up?
it is saved to the cloud. |
| 4.3 How will you test whether you can restore from your backups?
By getting the data from cloud. |

5. Sharing your data

| |
|--|
| 5.1 Who owns the data you generate?
I own the data. |
| 5.2 Who else has a right to see or use this data?
My supervisors has the right. |

| |
|--|
| 5.3 Who else should reasonably have access to this data when you share it? |
| Supervisors. |
| 5.4 What should/shouldn't be shared and why? |
| Personal details shouldn't be shared. |

6. Archiving your data

| |
|--|
| 6.1 What should be archived beyond the end of your project? |
| the code files have to be archived. |
| 6.2 For how long should it be stored? |
| Saved for 2 years. |
| 6.3 When will files be moved into the data archive/repository? |
| When it is completed. |
| 6.4 Where will the data be stored? |
| The data is stored in drive. |
| 6.5 Who is responsible for moving data to the data archive and maintaining it? |
| I am responsible |
| 6.6 Who should have access and under what conditions? |
| My supervisor |

7. Executing your plan

| |
|--|
| 7.1 Who is responsible for making sure this plan is followed? |
| I am responsible. |
| 7.2 How often will this plan be reviewed and updated? |
| Every day. |
| 7.3 What actions have you identified from the rest of this plan? |
| To improve the coding performance |
| 7.4 What further information do you need to carry out these actions? |
| Future research. |

• Author Contribution statement:

Sravani Ravipati designed the data collection. Nazeema Begum Rahman basha and Preethi Debli performed the data cleaning. Sravani Ravipati performed the exploratory data analysis. Nazeema begum Rahman basha has done with random forest with pyspark implementation.

APPENDIX:

1. CS5710 Week 7 Teaching Materials (2024). Lab7 : Pyspark-GoogleColab. Available at, accessed <https://brightspace.brunel.ac.uk/d2l/le/lessons/43719/topics/1534035> 19 February 2024.