**Bahria University,**

Karachi Campus



## LAB EXPERIMENT NO.

## 4

## LIST OF TASKS

|  |  |
| --- | --- |
| **TASK NO** | **OBJECTIVE** |
| **1** | Using python implement Decision Tree Algorithm on Diabetes Dataset the chances of diabetes in a person. visualize the results of the model in the form of a confusion matrix using matplotlib and seaborn. |
| **2** | Using Knime implement Task # 01. |
| **3** | Using python perform the parameter tuning to optimize the Decision Tree performance and compare the results with task # 1. |

**Submitted On:**

18 feb 2024

(Date: DD/MM/YY)

**TASK 1: Using python implement Decision Tree Algorithm on Diabetes Dataset the chances of diabetes in a person. visualize the results of the model in the form of a confusion matrix using matplotlib and seaborn.**

**SOLUTION:**

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import confusion\_matrix, accuracy\_score

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

import pandas as pd

file\_path = 'diabetes.csv'

diabetes\_data = pd.read\_csv(file\_path)

columns\_to\_fix = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']

for col in columns\_to\_fix:

diabetes\_data[col] = diabetes\_data[col].replace(0, diabetes\_data[col].median())

X = diabetes\_data.drop('Outcome', axis=1)

y = diabetes\_data['Outcome']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

dt\_classifier = DecisionTreeClassifier(random\_state=42)

dt\_classifier.fit(X\_train\_scaled, y\_train)

y\_pred = dt\_classifier.predict(X\_test\_scaled)

accuracy = accuracy\_score(y\_test, y\_pred)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues')

plt.title('Confusion Matrix for Decision Tree Model')

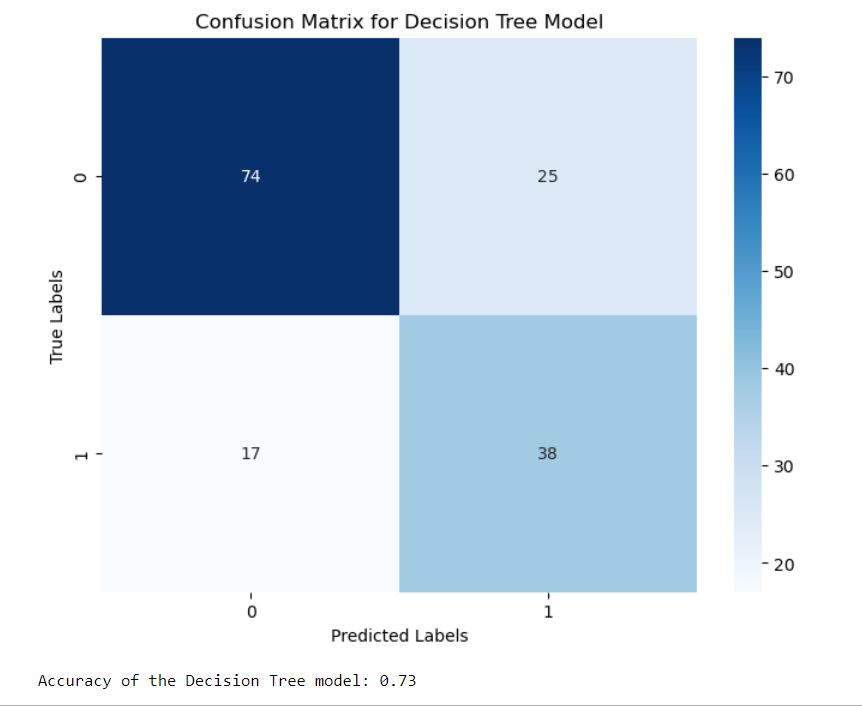
plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.show()

print(f"Accuracy of the Decision Tree model: {accuracy:.2f}")

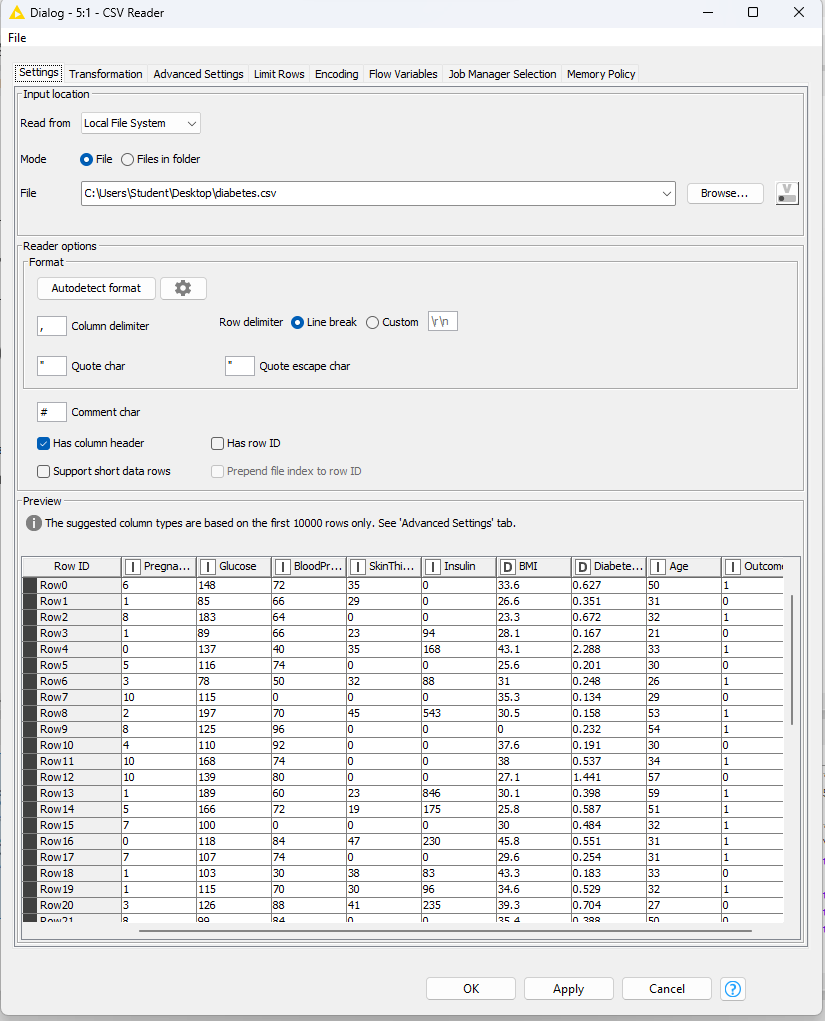
**OUTPUT:**

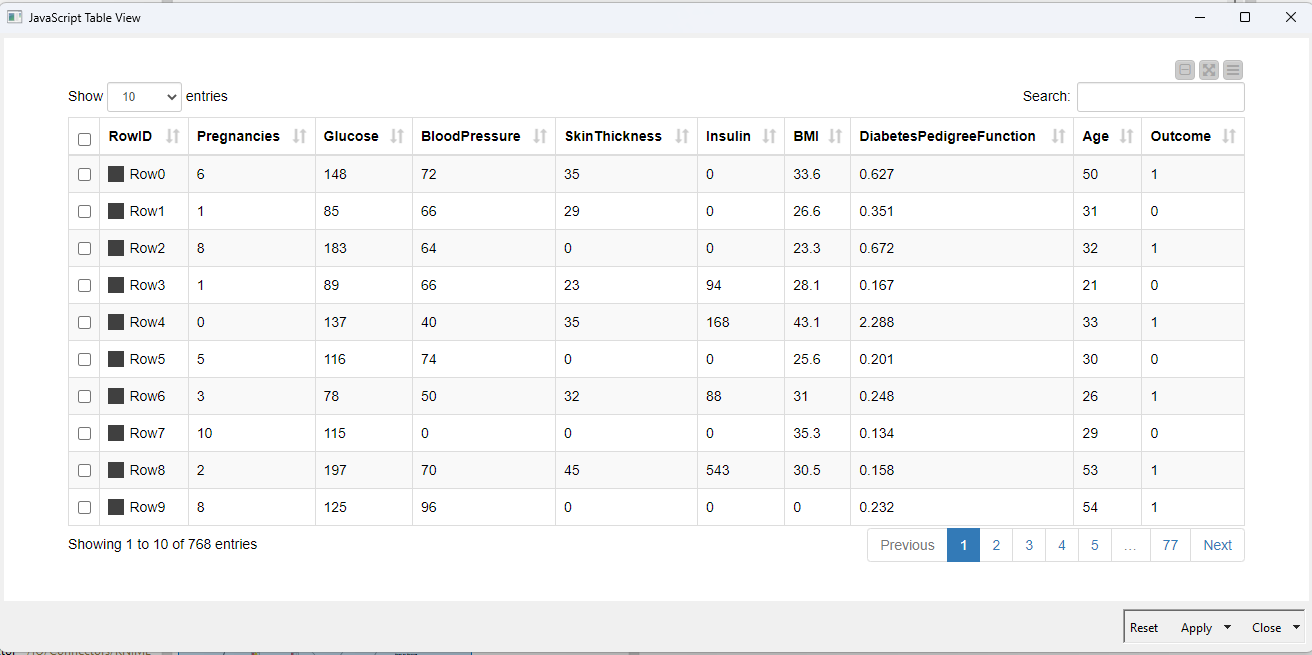
****

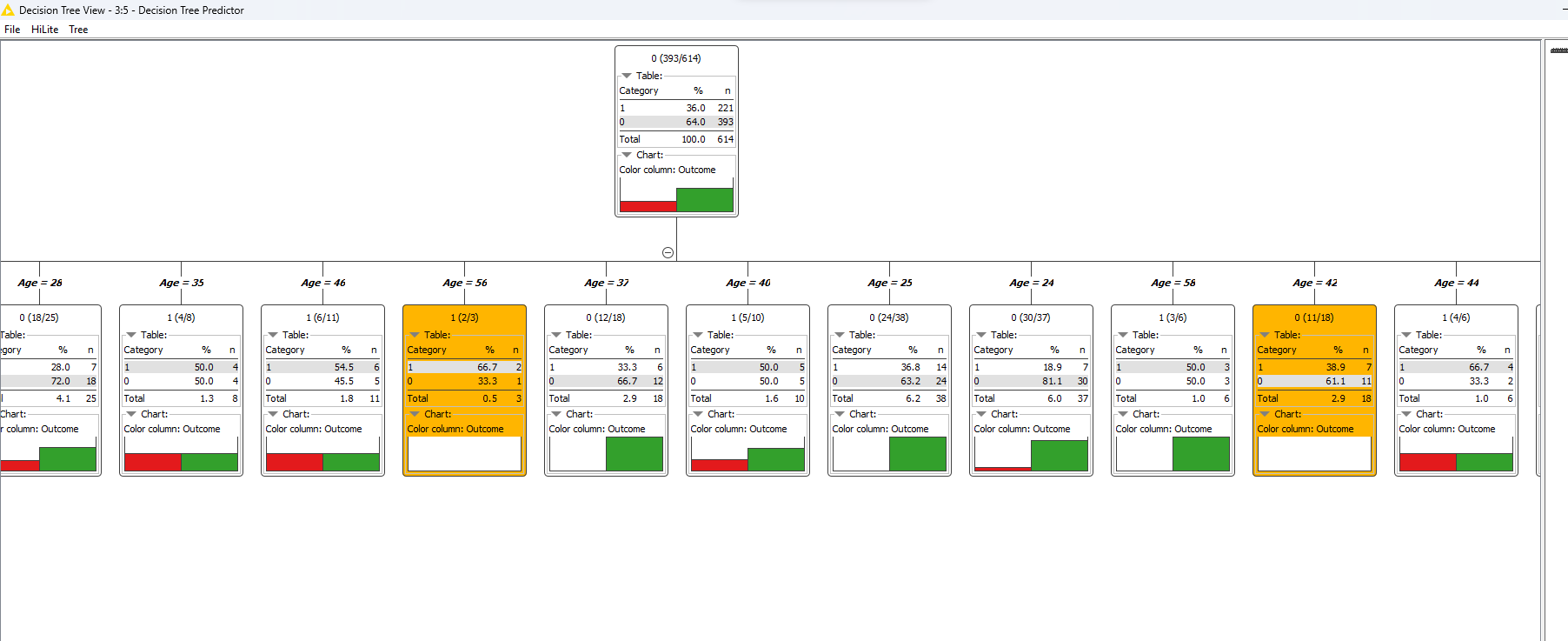
**TASK 2:** **Using Knime implement Task # 01.**

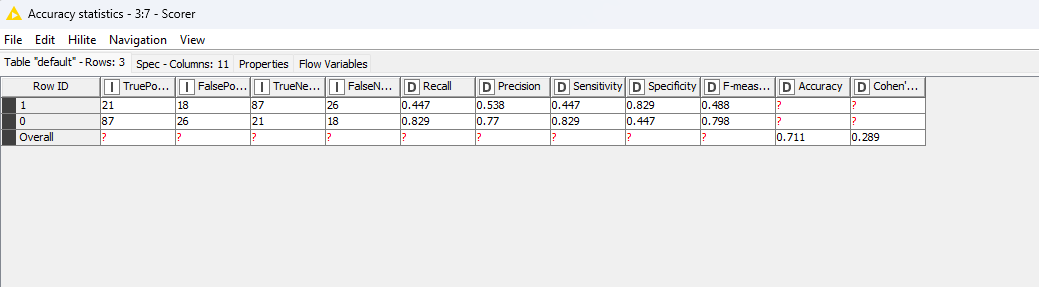
A screenshot of a computer

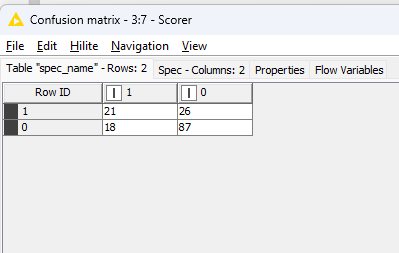
Description automatically generated

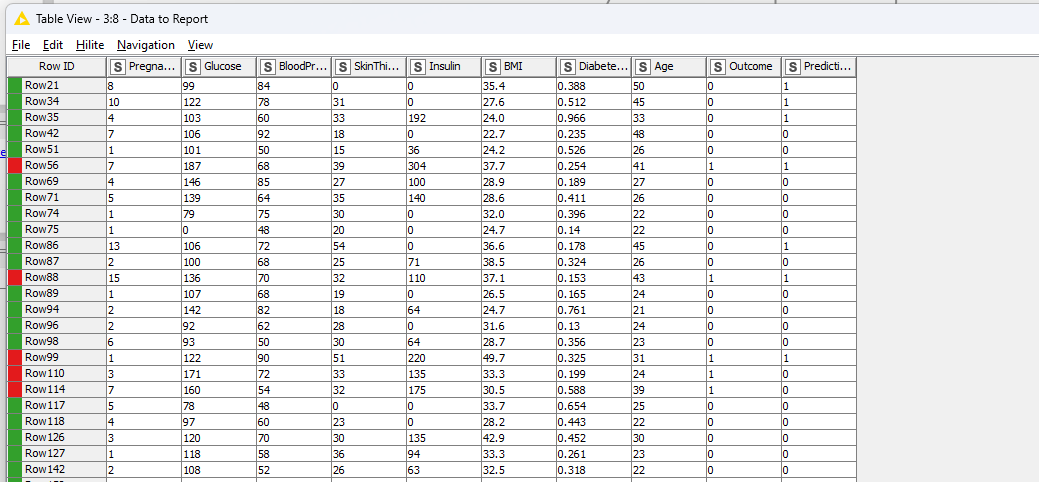












**TASK 3: Using python perform the parameter tuning to optimize the Decision Tree performance and compare the results with task # 1.**

**SOLUTION:**

from sklearn.model\_selection import GridSearchCV

param\_grid = {

'criterion': ['gini', 'entropy'],

'max\_depth': [None, 10, 20, 30, 40, 50],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4]

}

grid\_search = GridSearchCV(DecisionTreeClassifier(random\_state=42), param\_grid, cv=5, scoring='accuracy', n\_jobs=-1)

grid\_search.fit(X\_train\_scaled, y\_train)

best\_params = grid\_search.best\_params\_

print(f"Best parameters: {best\_params}")

optimized\_dt = DecisionTreeClassifier(\*\*best\_params, random\_state=42)

optimized\_dt.fit(X\_train\_scaled, y\_train)

y\_pred\_optimized = optimized\_dt.predict(X\_test\_scaled)

accuracy\_optimized = accuracy\_score(y\_test, y\_pred\_optimized)

confusion\_mtx\_optimized = confusion\_matrix(y\_test, y\_pred\_optimized)

plt.figure(figsize=(8, 6))

sns.heatmap(confusion\_mtx\_optimized, annot=True, fmt='d', cmap='Blues')

plt.title('Confusion Matrix for Optimized Decision Tree Model')

plt.xlabel('Predicted labels')

plt.ylabel('True labels')

plt.show()

print(f"Accuracy before optimization: {accuracy:.2f}")

print(f"Accuracy after optimization: {accuracy\_optimized:.2f}")

accuracy\_improvement = accuracy\_optimized - accuracy

accuracy\_improvement

**OUTPUT:**

