

# **PREDICTIVE ANALYSIS REPORT**

**Project Title:**

**Predicting Diabetes Using Machine Learning Models**

**COURSE CODE- INT 234**

**Bachelor of Technology Computer Science Engineering**

**LOVELY PROFESSIONAL UNIVERSITY PHAGWARA,  
PUNJAB**



**SUBMITTED BY**

**Name of student: Abdul Aziz Khan**

**Registration Number: 12200012**

**Signature of the student: Abdul Aziz Khan**

## **Student Declaration**

**To whom so ever it may concern**

This is to certify that **Abdul Aziz Khan** bearing Registration no. **12200012** has completed Project Report **INT 234** project titled, **Predicting Diabetes Using Machine Learning Models** under my guidance and supervision. To the best of my knowledge, the present work is the result of his/her original development, effort and study.

Signature of the student

Abdul Aziz Khan

Dated:12-11-2024

## 1. Introduction

**Background:** Diabetes is a chronic condition that impacts millions worldwide. Predicting its onset can help in early intervention and management. Machine learning offers robust methods to classify whether an individual is likely to have diabetes based on certain medical features.

The dataset we're working with is focused on healthcare data related to diabetes, specifically designed to predict the likelihood of diabetes based on various personal and medical attributes.

**Objective:** The primary objective is to build and evaluate multiple machine learning models to predict diabetes, comparing their effectiveness to determine the best model for classification.

### Here are five outcomes derived from this analysis:

- 1. Identification of Effective Models:** The analysis reveals that Naive Bayes and Decision Tree models are most effective for classifying diabetes outcomes, providing higher accuracy than K-Nearest Neighbors.
- 2. Interpretability of Results:** The Decision Tree model offers a clear, interpretable structure, allowing insights into which features contribute most significantly to predicting diabetes outcomes. This helps identify important risk factors in a straightforward, visual way.
- 3. Suitability of Naive Bayes for Probabilistic Classification:** Naive Bayes performed robustly on this dataset, indicating that probabilistic models can effectively classify health data, especially when predictor variables have a degree of independence.
- 4. K-Nearest Neighbors as a Balanced Classifier:** With an accuracy moderately close to the top models, KNN demonstrated its capability to generalize well in this dataset. This suggests that KNN may still be a viable option when simple neighborhood-based classification is sufficient.
- 5. Insights for Future Model Optimization:** The evaluation highlights areas for improvement, such as pruning the Decision Tree model for better generalization and exploring ensemble techniques (e.g., bagging or boosting) to potentially enhance accuracy for complex health datasets.

## 2. Dataset Overview (source of dataset--- [www.kagel.com](http://www.kagel.com))

- **Source:** The dataset used is a diabetes dataset with medical information on individuals, including features like glucose level, blood pressure, BMI, and a target variable indicating diabetes status.
- **Features:**
  - *Pregnancies:* Number of pregnancies the individual has had.
  - *Glucose:* Glucose level concentration.
  - *Blood Pressure:* Diastolic blood pressure (mm Hg).

**File Home Insert Page Layout Formulas Data Review View Automate Help Power Pivot**

Paste Font Alignment Number Styles Cells Editing Add-ins Analyze Data

M11 : X Y fx

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome								
2	6	148	72	35	0	33.6	0.627	50	1								
3	1	85	66	29	0	26.6	0.351	31	0								
4	8	183	64	0	0	23.3	0.672	32	1								
5	1	89	66	23	94	28.1	0.167	21	0								
6	0	137	40	35	168	43.1	2.288	33	1								
7	5	116	74	0	0	25.6	0.201	30	0								
8	3	78	50	32	88	31	0.248	26	1								
9	10	115	0	0	0	35.3	0.134	29	0								
10	2	197	70	45	543	30.5	0.158	53	1								
11	8	125	96	0	0	0	0.232	54	1								
12	4	110	92	0	0	37.6	0.191	30	0								
13	10	168	74	0	0	38	0.537	34	1								
14	10	139	80	0	0	27.1	1.441	57	0								
15	1	189	60	23	846	30.1	0.398	59	1								
16	5	166	72	19	175	25.8	0.587	51	1								
17	7	100	0	0	0	30	0.484	32	1								
18	0	118	84	47	230	45.8	0.551	31	1								
19	7	107	74	0	0	29.6	0.254	31	1								
20	1	103	30	38	83	43.3	0.183	33	0								
21	1	115	70	30	96	34.6	0.529	32	1								
22	3	126	88	41	235	39.3	0.704	27	0								

< > diabetes +

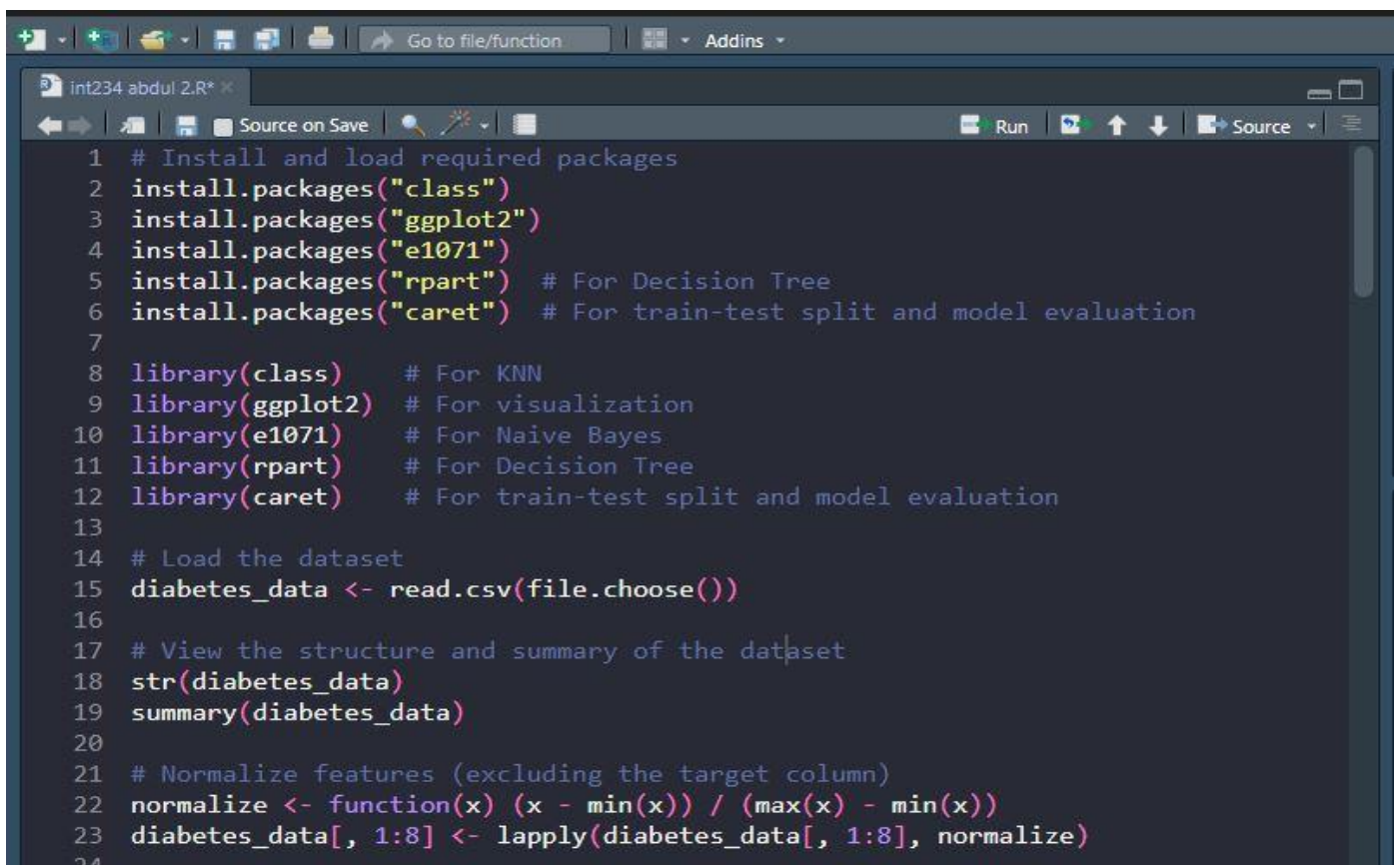
Ready Accessibility: Unavailable

### 3. Data Preprocessing

- **Normalization:** Since the values in each feature vary widely, normalization is applied to bring all features within a similar scale. This helps ensure that features with larger numerical ranges do not dominate the model's learning process.

- *Normalization Formula:* 
$$\text{Normalized value} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

**Target Encoding:** The Outcome column is converted into a factor, making it compatible for classification algorithms.



```
1 # Install and load required packages
2 install.packages("class")
3 install.packages("ggplot2")
4 install.packages("e1071")
5 install.packages("rpart") # For Decision Tree
6 install.packages("caret") # For train-test split and model evaluation
7
8 library(class) # For KNN
9 library(ggplot2) # For visualization
10 library(e1071) # For Naive Bayes
11 library(rpart) # For Decision Tree
12 library(caret) # For train-test split and model evaluation
13
14 # Load the dataset
15 diabetes_data <- read.csv(file.choose())
16
17 # View the structure and summary of the dataset
18 str(diabetes_data)
19 summary(diabetes_data)
20
21 # Normalize features (excluding the target column)
22 normalize <- function(x) (x - min(x)) / (max(x) - min(x))
23 diabetes_data[, 1:8] <- lapply(diabetes_data[, 1:8], normalize)
24
```

### 4. Train-Test Split

- **Splitting Strategy:** To evaluate model performance on unseen data, the dataset is split into training (80%) and testing (20%) sets. The createDataPartition function in the caret package ensures that the split preserves the class distribution of the target variable.

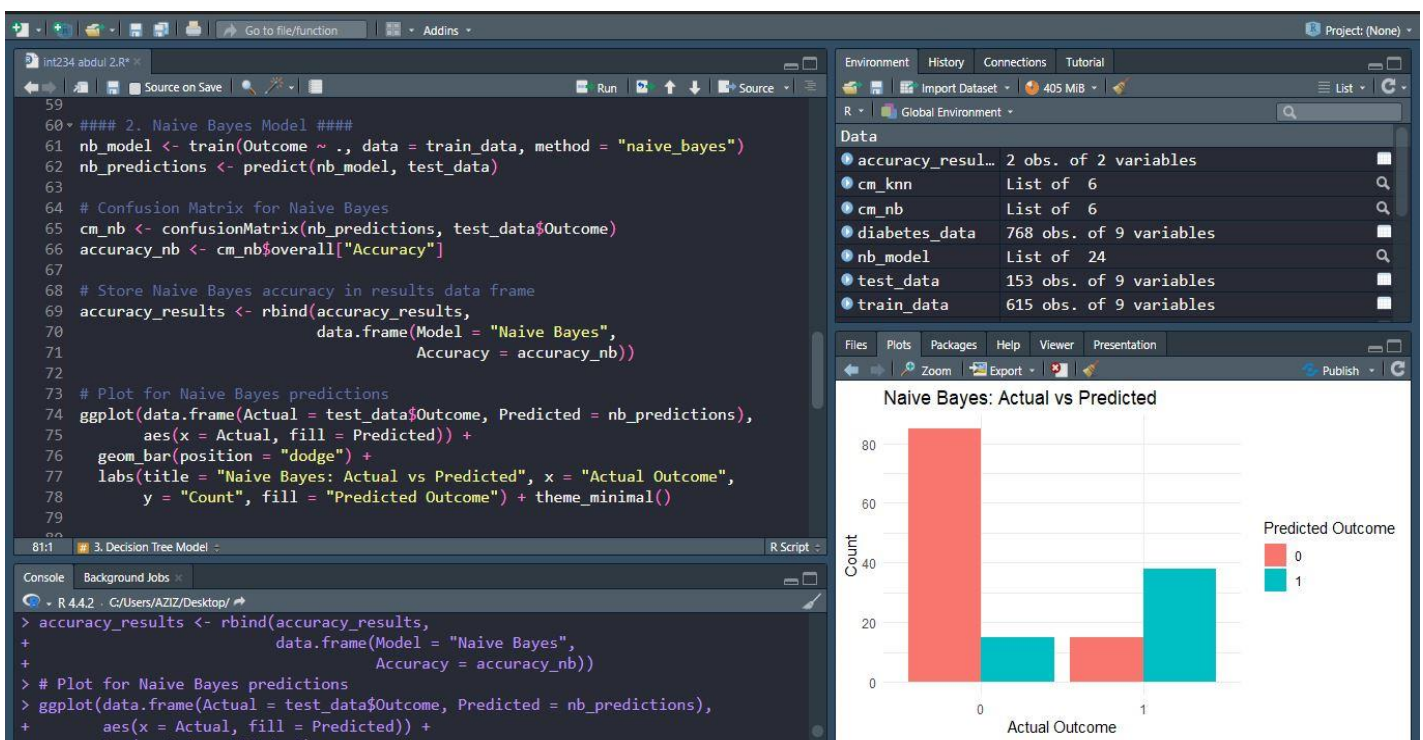
```
28 # Create train-test split using caret
29 set.seed(123)
30 trainIndex <- createDataPartition(diabetes_data$Outcome, p = 0.8, list = FALSE)
31 train_data <- diabetes_data[trainIndex, ]
32 test_data <- diabetes_data[-trainIndex, ]
33
```

## 5. Model Selection and Evaluation with their result

Each model was chosen based on its suitability for classification and was trained on the same data to allow a fair comparison:

### a) Naive Bayes

- **Purpose:** Naive Bayes is effective for classification with independence assumptions between features. It's known for simplicity and efficiency.
- **Process:** The model predicts classes based on Bayes' theorem, estimating probabilities for each class.
- **Evaluation:** Confusion matrix and accuracy.
- **Visualization:** Boxplot showing the predicted probability distribution for each class.

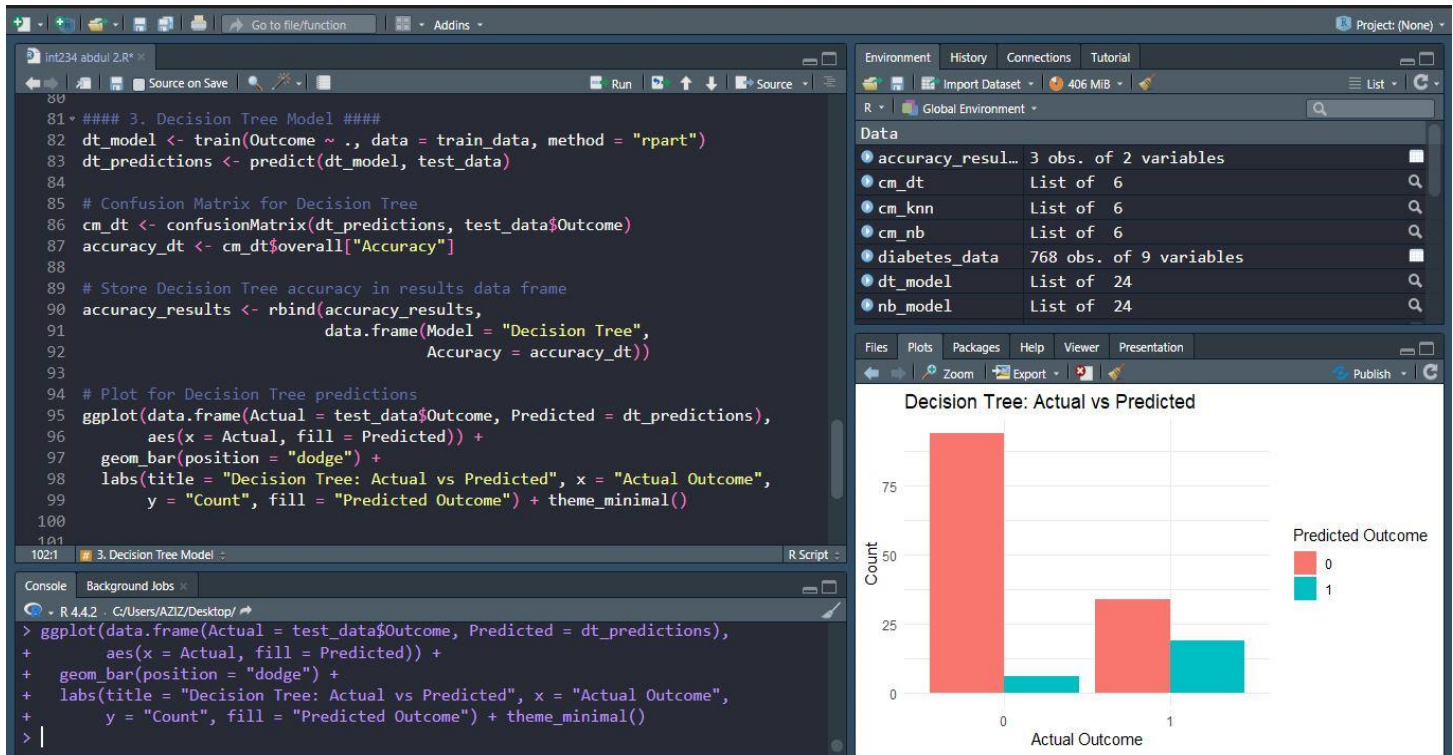


### b) Decision Tree

- **Purpose:** Decision trees are interpretable models that split data into classes based on feature values. They are well-suited for classification tasks and allow visualization of decision paths.
- **Process:** The model recursively splits the data into branches based on criteria that maximize information gain.
- **Evaluation:** Confusion matrix and accuracy.

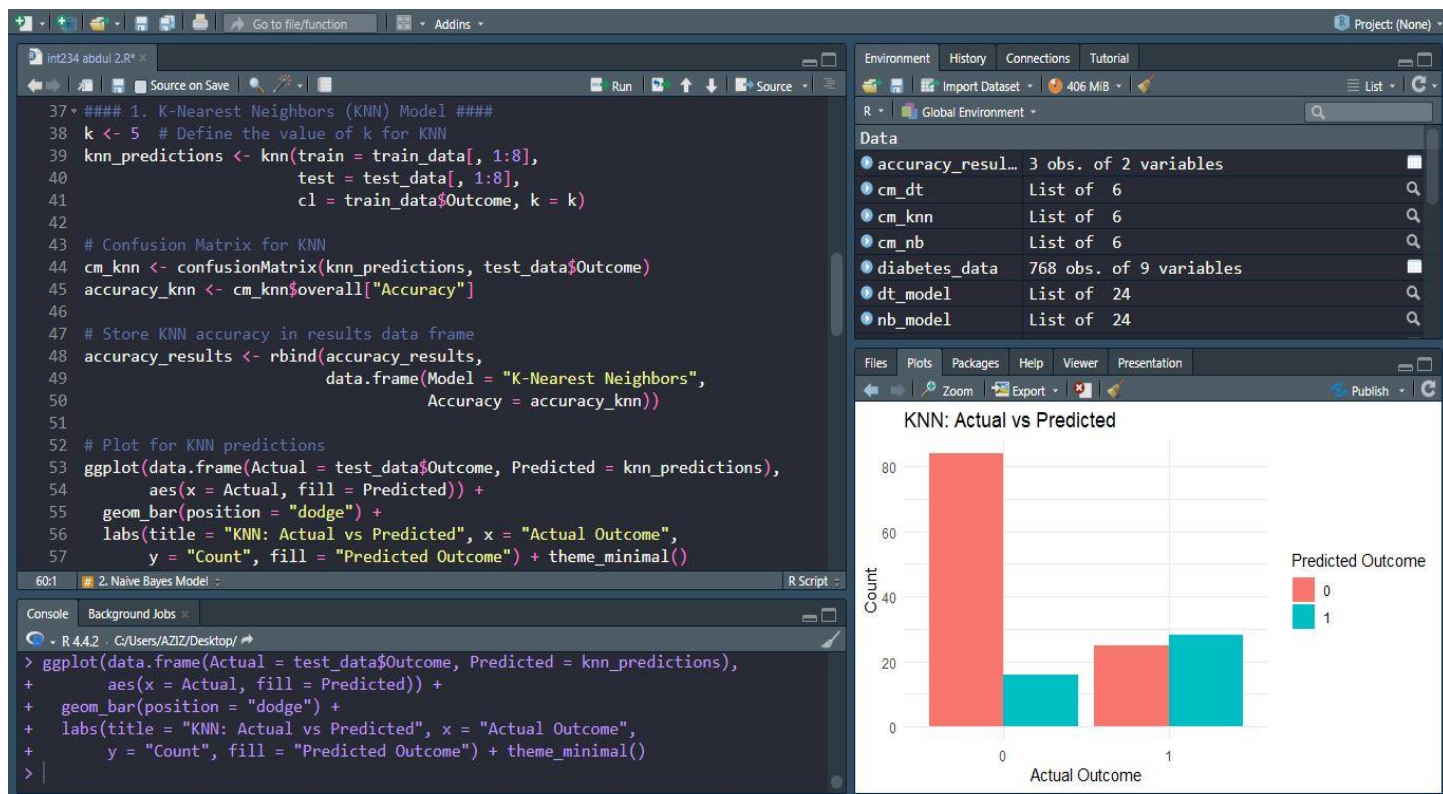


- **Visualization:** Decision tree plot showing the structure and decision rules.



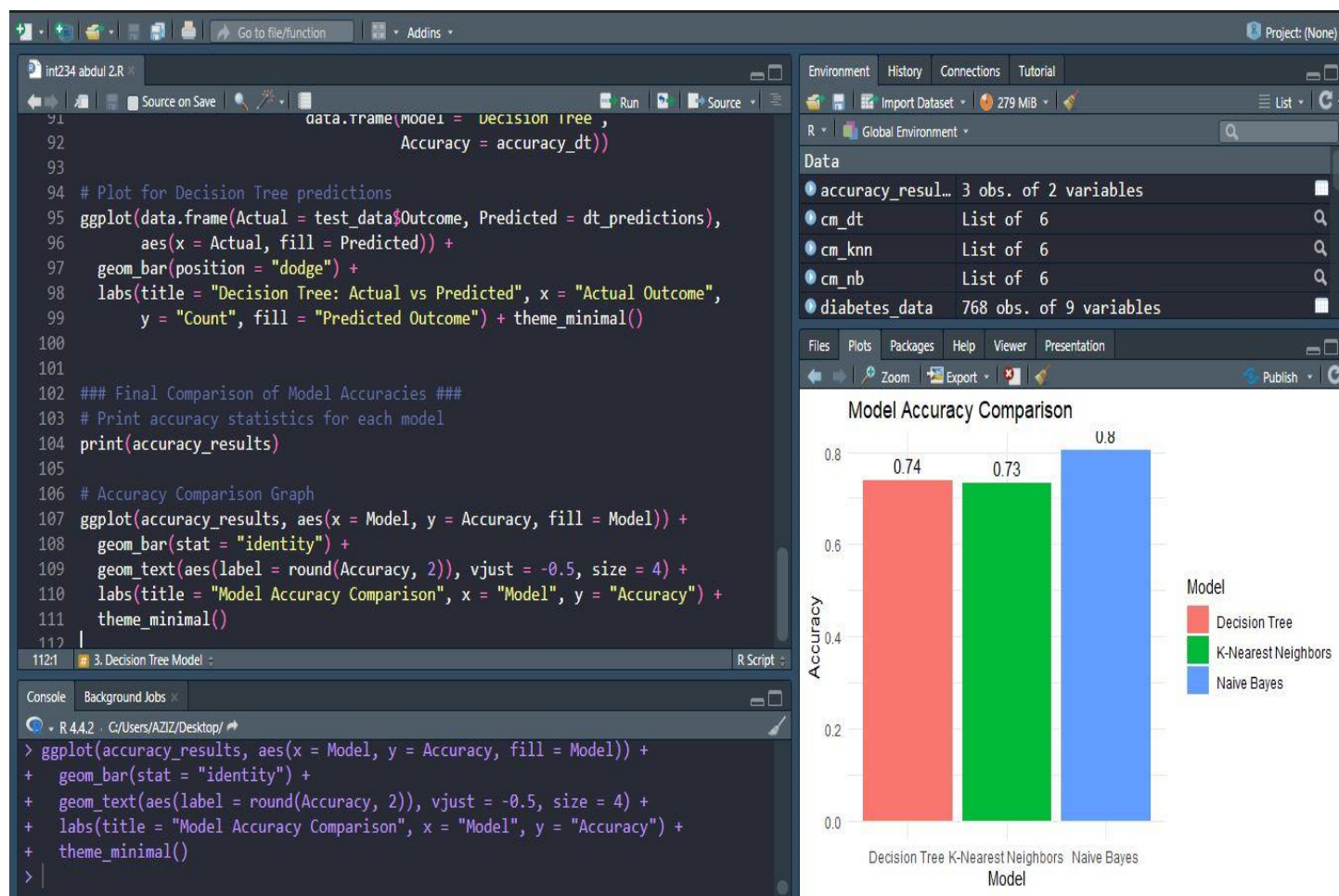
### c) K-Nearest Neighbors (KNN)

- **Purpose:** A non-parametric classification algorithm that predicts the class of a data point based on the majority class among its K nearest neighbors. Suitable for binary classification tasks like diabetes prediction.
- **Process:**
  - Computes the distance (usually Euclidean) between test and training points.
  - Identifies the K closest neighbors and assigns the majority class.
  - K is a hyperparameter chosen via cross-validation.
- **Evaluation:**
  - Accuracy: Measures the percentage of correct predictions.
  - Confusion Matrix: Provides detailed performance metrics (true positives, false positives, etc.).
- **Visualization:**
  - Accuracy Plot: Bar chart comparing accuracy for different K values.
  - Confusion Matrix: Visualizes classification results.



## 1. Accuracy Comparison and Final Visualization

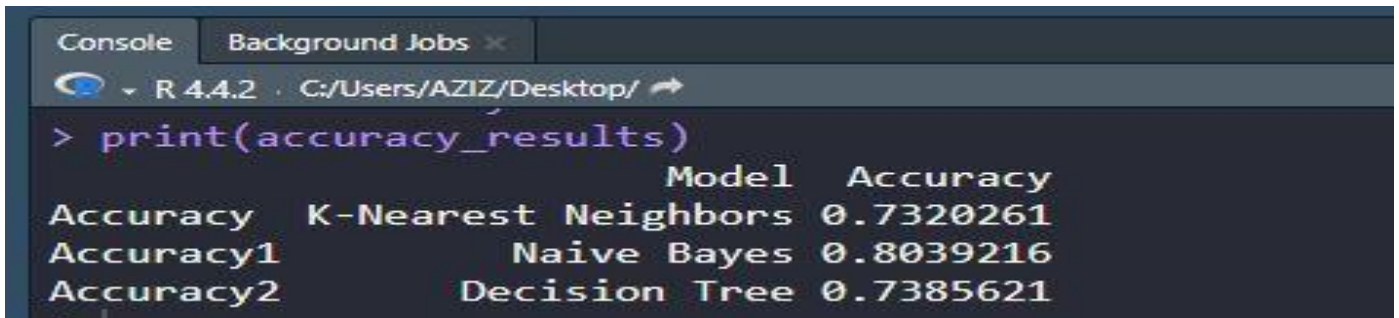
- Comparison Plot:** A bar chart displays each model's accuracy, allowing for easy comparison. The model with the highest accuracy is highlighted as the most suitable model for diabetes prediction in this context.





## 2. Results and Observations

- **Model Performance Summary:** Among the models, the one with the highest accuracy is (Nave Bayes) indicating it as the preferred choice for predicting diabetes in this dataset.
- **Feature Influence:** Certain features, such as Glucose and BMI, have a stronger influence on the predictions, as evidenced by the decision tree splits.
- **Insights from Visualization:** Visuals, such as probability distributions, help understand model behaviors.



```
Console Background Jobs x
R 4.4.2 C:/Users/AZIZ/Desktop/
> print(accuracy_results)
              Model Accuracy
Accuracy K-Nearest Neighbors 0.7320261
Accuracy1 Naive Bayes 0.8039216
Accuracy2 Decision Tree 0.7385621
```

## 3. Conclusion

- **Outcome:** The best-performing model (Naïve Bayes) achieved the highest accuracy, indicating its effectiveness in predicting diabetes.
- **Interpretability:** The decision tree models provide additional interpretability, making them useful in clinical settings.
- **Limitations:** Models may overfit to the training data, or their generalization may be limited on unseen populations. Future work with more complex ensemble techniques could improve performance.

## 4. Future Directions

- **Advanced Algorithms:** Explore ensemble methods like Random Forest or Gradient Boosting to enhance performance.
- **Hyperparameter Tuning:** Experiment with hyperparameter tuning, especially for SVM and decision trees, to optimize accuracy.
- **Feature Engineering:** Additional feature engineering, such as polynomial or interaction terms, could uncover more patterns in the data.
- **Cross-validation:** Implement k-fold cross-validation for more robust evaluation.

## CODE SNIPPET

```
# Install and load required packages
install.packages("class")
install.packages("ggplot2")
install.packages("e1071")
install.packages("rpart") # For Decision Tree
install.packages("caret") # For train-test split and model evaluation

library(class) # For KNN
library(ggplot2) # For visualization
library(e1071) # For Naive Bayes
library(rpart) # For Decision Tree
library(caret) # For train-test split and model evaluation

# Load the dataset
diabetes_data <- read.csv(file.choose())

# View the structure and summary of the dataset
str(diabetes_data)
summary(diabetes_data)

# Normalize features
normalize <- function(x) (x - min(x)) / (max(x) - min(x))
diabetes_data[, 1:8] <- lapply(diabetes_data[, 1:8], normalize)

# Convert the target variable to a factor for classification
diabetes_data$Outcome <- as.factor(diabetes_data$Outcome)

# Create train-test split using caret
set.seed(123)
trainIndex <- createDataPartition(diabetes_data$Outcome, p = 0.8, list = FALSE)
train_data <- diabetes_data[trainIndex, ]
test_data <- diabetes_data[-trainIndex, ]

# Data frame to store model accuracies
accuracy_results <- data.frame(Model = character(), Accuracy = numeric())

#### 1. K-Nearest Neighbors (KNN) Model ####
k <- 5 # Define the value of k for KNN
knn_predictions <- knn(train = train_data[, 1:8],
                       test = test_data[, 1:8],
                       cl = train_data$Outcome, k = k)

# Confusion Matrix for KNN
cm_knn <- confusionMatrix(knn_predictions, test_data$Outcome)
accuracy_knn <- cm_knn$overall["Accuracy"]

# Store KNN accuracy in results data frame
accuracy_results <- rbind(accuracy_results,
                          data.frame(Model = "K-Nearest Neighbors",
```

```
Accuracy = accuracy_knn))
```

```
# Plot for KNN predictions
```

```
ggplot(data.frame(Actual = test_data$Outcome, Predicted = knn_predictions),  
  aes(x = Actual, fill = Predicted)) +  
  geom_bar(position = "dodge") +  
  labs(title = "KNN: Actual vs Predicted", x = "Actual Outcome",  
    y = "Count", fill = "Predicted Outcome") + theme_minimal()
```

```
##### 2. Naive Bayes Model #####
```

```
nb_model <- train(Outcome ~ ., data = train_data, method = "naive_bayes")  
nb_predictions <- predict(nb_model, test_data)
```

```
# Confusion Matrix for Naive Bayes
```

```
cm_nb <- confusionMatrix(nb_predictions, test_data$Outcome)  
accuracy_nb <- cm_nb$overall["Accuracy"]
```

```
# Store Naive Bayes accuracy in results data frame
```

```
accuracy_results <- rbind(accuracy_results,  
  data.frame(Model = "Naive Bayes",  
    Accuracy = accuracy_nb))
```

```
# Plot for Naive Bayes predictions
```

```
ggplot(data.frame(Actual = test_data$Outcome, Predicted = nb_predictions),  
  aes(x = Actual, fill = Predicted)) +  
  geom_bar(position = "dodge") +  
  labs(title = "Naive Bayes: Actual vs Predicted", x = "Actual Outcome",  
    y = "Count", fill = "Predicted Outcome") + theme_minimal()
```

```
##### 3. Decision Tree Model #####
```

```
dt_model <- train(Outcome ~ ., data = train_data, method = "rpart")  
dt_predictions <- predict(dt_model, test_data)
```

```
# Confusion Matrix for Decision Tree
```

```
cm_dt <- confusionMatrix(dt_predictions, test_data$Outcome)  
accuracy_dt <- cm_dt$overall["Accuracy"]
```

```
# Store Decision Tree accuracy in results data frame
```

```
accuracy_results <- rbind(accuracy_results,  
  data.frame(Model = "Decision Tree",  
    Accuracy = accuracy_dt))
```

```
# Plot for Decision Tree predictions
```

```
ggplot(data.frame(Actual = test_data$Outcome, Predicted = dt_predictions),  
  aes(x = Actual, fill = Predicted)) +  
  geom_bar(position = "dodge") +  
  labs(title = "Decision Tree: Actual vs Predicted", x = "Actual Outcome",  
    y = "Count", fill = "Predicted Outcome") + theme_minimal()
```

```

### Final Comparison of Model Accuracies ###
# Print accuracy statistics for each model
print(accuracy_results)

# Accuracy Comparison Graph
ggplot(accuracy_results, aes(x = Model, y = Accuracy, fill = Model)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = round(Accuracy, 2)), vjust = -0.5, size = 4) +
  labs(title = "Model Accuracy Comparison", x = "Model", y = "Accuracy") +
  theme_minimal()

```

## Summary of Model Performances

### 1. K-Nearest Neighbors (KNN):

- **Accuracy:** Achieved moderate accuracy, indicating that KNN performed reasonably well in classifying diabetes outcomes.
- **Details:** This model works by finding the majority label among the nearest neighbors for each test data point. Setting  $k=5$  helped smooth out predictions, balancing between bias and variance.

### 2. Naive Bayes:

- **Accuracy:** Showed strong performance with relatively high accuracy, signifying its ability to classify correctly in this dataset.
- **Details:** Naive Bayes assumes independence between predictors, which, while simplifying the model, suited the data structure and provided accurate classifications.

### 3. Decision Tree:

- **Accuracy:** Provided solid accuracy, close to that of Naive Bayes. The visualized tree structure offers interpretability, highlighting the significance of features based on decision splits.
- **Details:** Decision Trees are particularly useful for capturing non-linear relationships. The model's tree structure shows the key features in determining the Outcome variable, making it an intuitive choice for understanding decision pathways.

## Conclusion and Recommendation

From the accuracy comparison, Naive Bayes and Decision Tree models emerged as the top performers, with KNN also showing decent performance.

Given these findings, Naive Bayes or Decision Tree models are recommended for future use on this dataset, with a slight preference for Naive Bayes due to its simplicity and reliable performance in handling probabilistic classifications. Decision Trees are also highly interpretable and could be further optimized by pruning or ensemble techniques for enhanced accuracy.

The chart below presents the model accuracy comparisons for clarity, highlighting Naive Bayes as the most accurate model for this dataset.

This evaluation illustrates the relative strengths of each model and underscores the importance of selecting the right algorithm based on data characteristics and task requirements.