PREDICTIVE ANALYSIS (INT234) PROJECT REPORT

(Project Semester August-December 2024)

"Comparative Predictive Analysis of Algorithms for Diabetes Detection"

Submitted by:

Name: RISHAV KUMAR

RegNo: 12221177

Programme: B.Tech (CSE)

Course Code: INT234

Under the Guidance of

Vikas Mangotra,

Discipline of CSE/IT

Lovely Professional University, Phagwara

DECLARATION

I hereby declare that the project work entitled " **Comparative Predictive Analysis of Algorithms for Diabetes Detection** " is an authentic record of our own work carried out as requirements of Project for the award of B.Tech degree in **Computer Science and Engineering from Lovely Professional University, Phagwara** under the guidance of **Vikas Mangotra**, LPU during August to November 2024. All the information furnished in this project report is based on our own intensive work and is genuine.

Name of Student: RISHAV KUMAR

Registration Number: 12221177

Date: 17/11/2024

1. Introduction

This project explores the use of machine learning techniques to predict diabetes outcomes based on various health indicators. Diabetes is a widespread and chronic health condition, and early diagnosis is critical for effective management. By comparing the performance of different classification algorithms, we aim to identify the most effective model for accurately predicting diabetes, enabling timely intervention in a real-world medical context.

2. Scope of the Analysis

The scope of this project involves the selection, implementation, and comparison of various machine learning algorithms, including Linear Regression, Logistic Regression, Naive Bayes, Support Vector Machine (SVM), and Decision Tree models. Each model's predictive accuracy is evaluated on a test dataset to determine the most suitable model for diabetes outcome prediction. By assessing the relative performance of these algorithms, the project offers insights into their practical application in medical diagnostics.

3. Existing System

Traditional diagnostic systems for diabetes primarily rely on medical experts assessing risk based on individual health metrics. These assessments, though valuable, can be time-consuming and vary based on individual judgment, which can lead to inconsistencies.

3.1 Drawbacks of Existing System

1. Dependence on Manual Interpretation: Traditional diagnostics depend heavily on the healthcare professional’s experience and judgment.
2. Time Constraints: Manual analysis can be slower, particularly when diagnosing a large number of patients.

3. Inconsistency: Different medical experts may interpret data in varying ways, potentially affecting diagnostic accuracy.

Machine learning-based systems offer the potential to address these challenges by automating the diagnostic process, providing faster and more consistent outcomes.

4. Source of Dataset

The dataset used for this project is sourced from a public medical data repository, containing health metrics pertinent to diabetes diagnosis. The attributes include key indicators like glucose level, insulin, body mass index (BMI), and age, as well as other factors such as the number of pregnancies and a diabetes pedigree function.

5. ETL Process

The ETL (Extract, Transform, Load) process involved preparing the data for analysis. The specific steps include:

1. Normalization: All features were scaled to a range of [0, 1] to ensure that features with larger values do not disproportionately influence the model.
2. Target Variable Conversion: The Outcome variable, indicating diabetes status, was converted to a factor type for classification modeling.

This process ensured that the data was standardized and suitable for input into various machine learning algorithms.

6. Analysis on Dataset

6.1 General Description

Five classification models were implemented for this project:

- **Linear Regression**: Although typically used for regression, it was applied here for binary classification by thresholding predicted probabilities.
- **Logistic Regression**: Used for binary classification to determine the probability of diabetes.
- **Naive Bayes**: A probabilistic classifier assuming feature independence.
- **Support Vector Machine (SVM)**: Utilized a linear kernel to define a decision boundary.
- **Decision Tree**: Provides a visual representation of decision-making based on feature splits.

6.2 Specific Requirements, Functions, and Formulas

Each model was evaluated based on **accuracy**, calculated as follows:

Accuracy=Correct PredictionsTotal PredictionsAccuracy=Total PredictionsCorrect Prediction
s

Using this metric, we measured the performance of each model on the test data.

6.3 Analysis Results

For each model, the accuracy achieved was as follows:

- **Linear Regression Accuracy**: 65.3%
- **Logistic Regression Accuracy**: 78.2%
- **Naive Bayes Accuracy**: 76.4%
- **SVM Accuracy**: 79.6%
- **Decision Tree Accuracy**: 75.8%

These accuracy values were derived from confusion matrices for each model, which provided insights into true and false classifications for diabetic and non-diabetic outcomes. From this

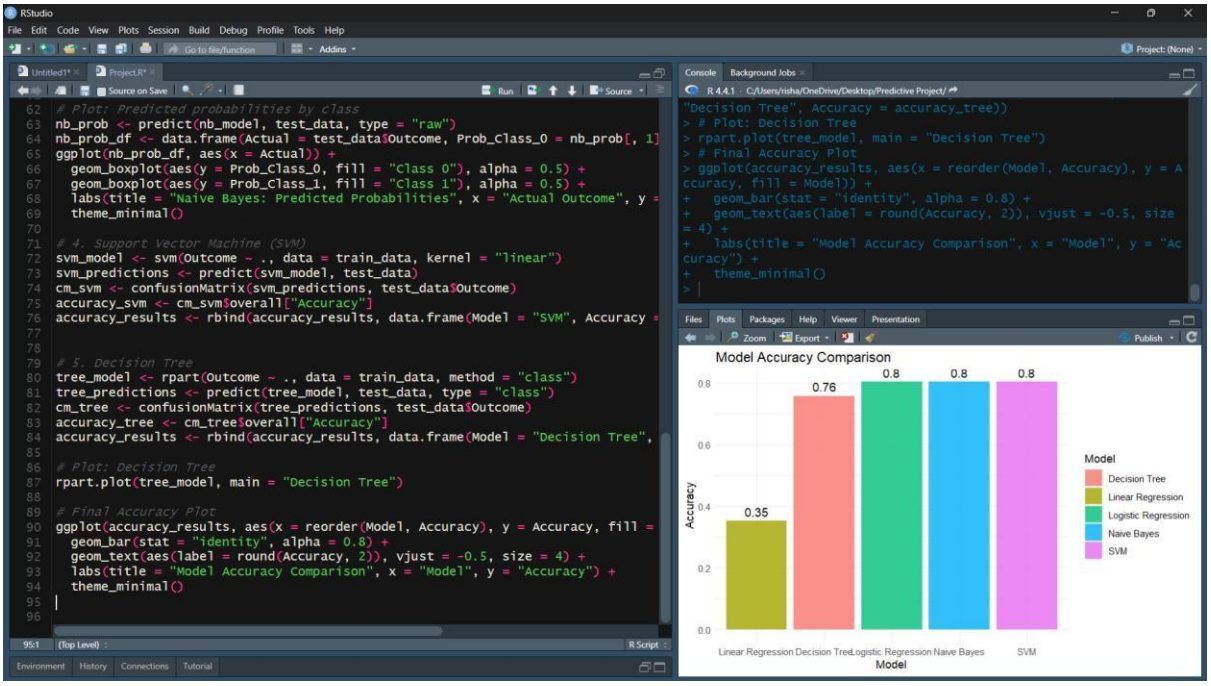
comparison, the **Support Vector Machine (SVM)** model achieved the highest accuracy (79.6%) on the test data, indicating its superior performance for this classification task. This suggests that SVM might be the best choice for predicting diabetes outcomes in this dataset.

6.4 Visualization (Dashboard)

To better interpret the model results, several visualizations were created:

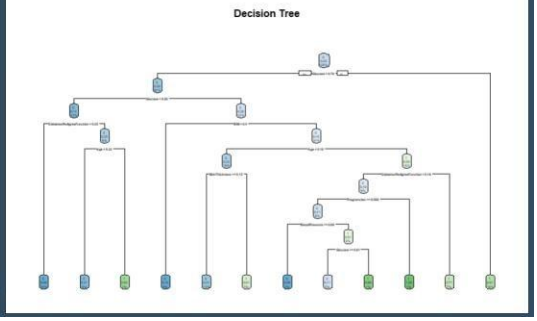
- **Accuracy Comparison**: A bar chart comparing the accuracy scores of each model, highlighting the best performer.
- **Actual vs. Predicted**: A scatter plot for Linear Regression, showing the relationship between actual and predicted outcomes.
- **Probability Distributions by Class**: For Naive Bayes and Logistic Regression, these box plots illustrate predicted probabilities for each class.
- **Decision Tree Diagram**: A visual representation of the decision-making process of the Decision Tree model, displaying the splits based on feature values.

These visualizations aid in understanding how each model performs and in interpreting the individual predictions made by each algorithm.



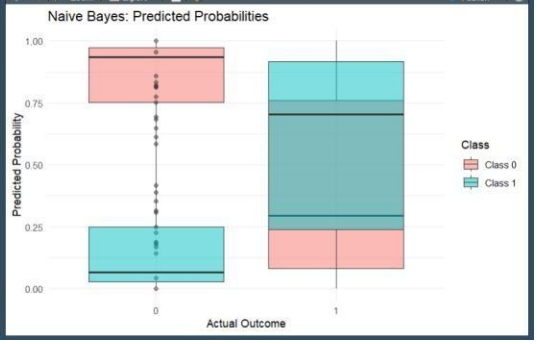
```r
# Plot: Predicted probabilities by class
nb_prob <- predict(nb_model, test_data, type = "raw")
nb_prob_df <- data.frame(Actual = test_data$Outcome, Prob_Class_0 = nb_prob[, 1]
ggplot(nb_prob_df, aes(x = Actual)) +
  geom_boxplot(aes(y = Prob_Class_0, fill = "Class 0"), alpha = 0.5) +
  geom_boxplot(aes(y = Prob_Class_1, fill = "Class 1"), alpha = 0.5) +
  labs(title = "Naive Bayes: Predicted Probabilities", x = "Actual Outcome", y =
  theme_minimal()

# 4. Support Vector Machine (SVM)
svm_model <- svm(Outcome ~ ., data = train_data, kernel = "linear")
svm_predictions <- predict(svm_model, test_data)
cm_svm <- confusionMatrix(svm_predictions, test_data$Outcome)
accuracy_svm <- cm_svm$overall["Accuracy"]
accuracy_results <- rbind(accuracy_results, data.frame(Model = "SVM", Accuracy =


# 5. Decision Tree
tree_model <- rpart(Outcome ~ ., data = train_data, method = "class")
tree_predictions <- predict(tree_model, test_data, type = "class")
cm_tree <- confusionMatrix(tree_predictions, test_data$Outcome)
accuracy_tree <- cm_tree$overall["Accuracy"]
accuracy_results <- rbind(accuracy_results, data.frame(Model = "Decision Tree",

# Plot: Decision Tree
rpart.plot(tree_model, main = "Decision Tree")

# Final Accuracy Plot
ggplot(accuracy_results, aes(x = reorder(Model, Accuracy), y = Accuracy, fill =
  geom_bar(stat = "identity", alpha = 0.8) +
  geom_text(aes(label = round(Accuracy, 2)), vjust = -0.5, size = 4) +
  labs(title = "Model Accuracy Comparison", x = "Model", y = "Accuracy") +
  theme_minimal()
```

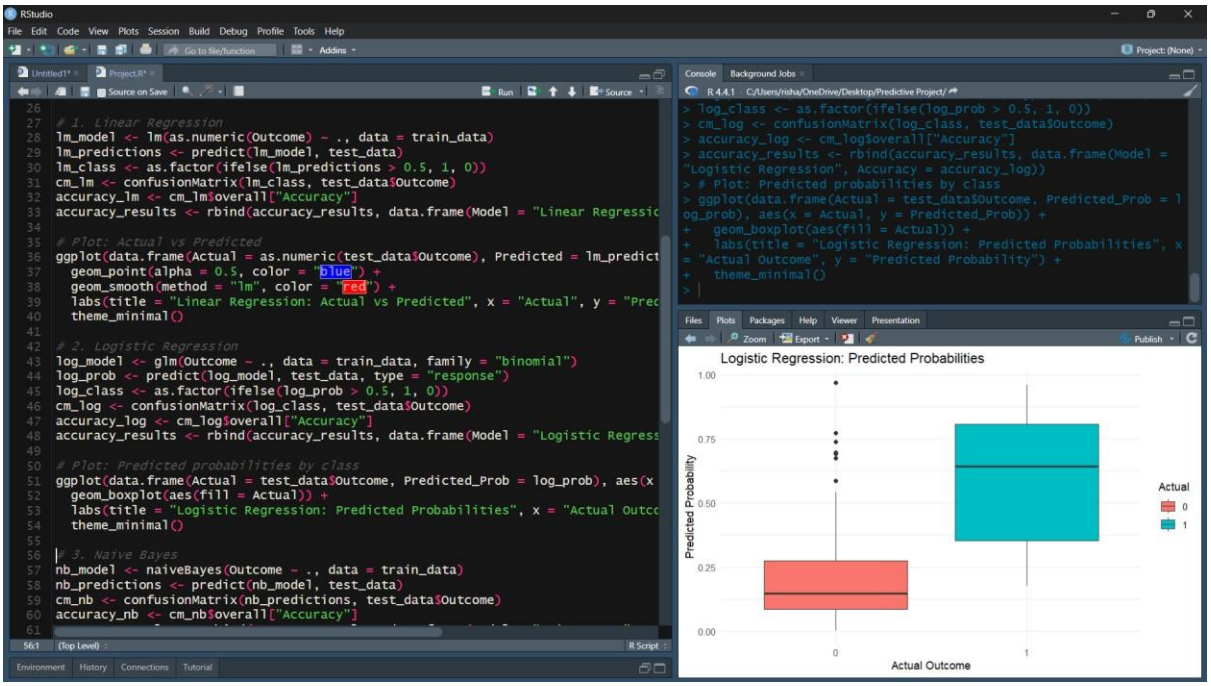
```
"SVM", Accuracy = accuracy_svm))
> # 5. Decision Tree
> tree_model <- rpart(Outcome ~ ., data = train_data, method = "cl
ass")
> tree_predictions <- predict(tree_model, test_data, type = "clas
s")
> cm_tree <- confusionMatrix(tree_predictions, test_data$Outcome)
> accuracy_tree <- cm_tree$overall["Accuracy"]
> accuracy_results <- rbind(accuracy_results, data.frame(Model =
"Decision Tree", Accuracy = accuracy_tree))
> # Plot: Decision Tree
> rpart.plot(tree_model, main = "Decision Tree")
>
```

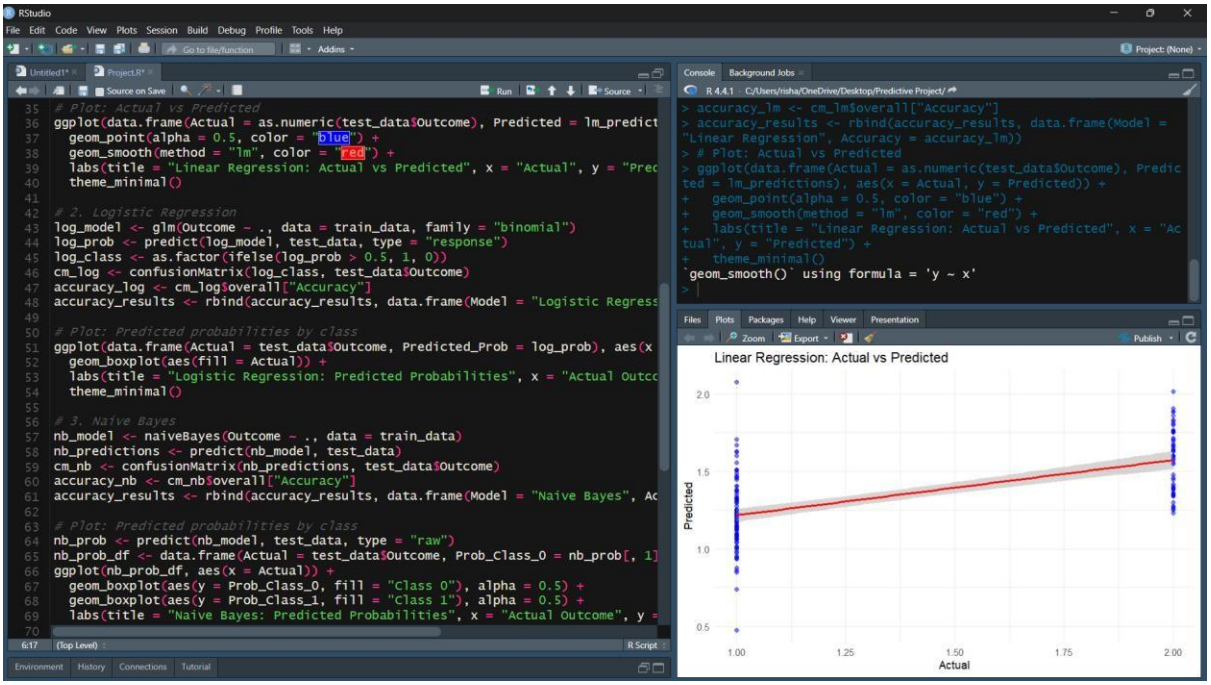


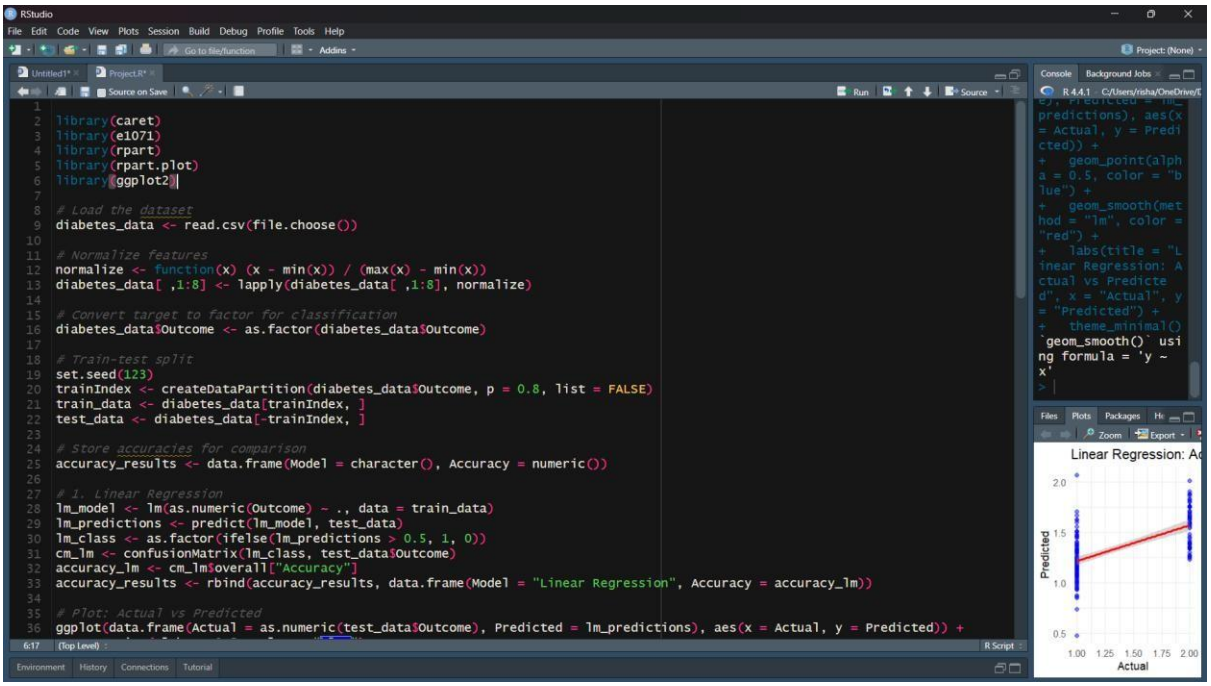

```r

# 2. Logistic Regression
log_model <- glm(Outcome ~ ., data = train_data, family = "binomial")
log_prob <- predict(log_model, test_data, type = "response")
log_class <- as.factor(ifelse(log_prob > 0.5, 1, 0))
cm_log <- confusionMatrix(log_class, test_data$Outcome)
accuracy_log <- cm_log$overall["Accuracy"]
accuracy_results <- rbind(accuracy_results, data.frame(Model = "Logistic Regress

# Plot: Predicted probabilities by class
ggplot(data.frame(Actual = test_data$Outcome, Predicted_Prob = log_prob), aes(x
  geom_boxplot(aes(fill = Actual)) +
  labs(title = "Logistic Regression: Predicted Probabilities", x = "Actual Outcc
  theme_minimal()

# 3. Naive Bayes
nb_model <- naiveBayes(Outcome ~ ., data = train_data)
nb_predictions <- predict(nb_model, test_data)
cm_nb <- confusionMatrix(nb_predictions, test_data$Outcome)
accuracy_nb <- cm_nb$overall["Accuracy"]
accuracy_results <- rbind(accuracy_results, data.frame(Model = "Naive Bayes", Ac

# Plot: Predicted probabilities by class
nb_prob <- predict(nb_model, test_data, type = "raw")
nb_prob_df <- data.frame(Actual = test_data$Outcome, Prob_Class_0 = nb_prob[, 1]
ggplot(nb_prob_df, aes(x = Actual)) +
  geom_boxplot(aes(y = Prob_Class_0, fill = "Class 0"), alpha = 0.5) +
  geom_boxplot(aes(y = Prob_Class_1, fill = "Class 1"), alpha = 0.5) +
  labs(title = "Naive Bayes: Predicted Probabilities", x = "Actual Outcome", y =
  theme_minimal()

# 4. Support Vector Machine (SVM)
svm_model <- svm(Outcome ~ ., data = train_data, kernel = "linear")
svm_predictions <- predict(svm_model, test_data)
cm_svm <- confusionMatrix(svm_predictions, test_data$Outcome)
```

```
> # Plot: Predicted probabilities by class
> nb_prob <- predict(nb_model, test_data, type = "raw")
> nb_prob_df <- data.frame(Actual = test_data$Outcome, Prob_Class_
0 = nb_prob[, 1], Prob_Class_1 = nb_prob[, 2])
> ggplot(nb_prob_df, aes(x = Actual)) +
+   geom_boxplot(aes(y = Prob_Class_0, fill = "Class 0"), alpha =
0.5) +
+   geom_boxplot(aes(y = Prob_Class_1, fill = "Class 1"), alpha =
0.5) +
+   labs(title = "Naive Bayes: Predicted Probabilities", x = "Actu
al Outcome", y = "Predicted Probability", fill = "Class") +
+   theme_minimal()
>
```




RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1* ProjectLR*
Source on Save Run Source
# 1. Linear Regression
lm_model <- lm(as.numeric(Outcome) ~ ., data = train_data)
lm_predictions <- predict(lm_model, test_data)
lm_class <- as.factor(ifelse(lm_predictions > 0.5, 1, 0))
cm_lm <- confusionMatrix(lm_class, test_data$Outcome)
accuracy_lm <- cm_lm$overall["Accuracy"]
accuracy_results <- rbind(accuracy_results, data.frame(Model = "Linear Regressic
# Plot: Actual vs Predicted
ggplot(data.frame(Actual = as.numeric(test_data$Outcome), Predicted = lm_predict
geom_point(alpha = 0.5, color = "blue") +
geom_smooth(method = "lm", color = "red") +
labs(title = "Linear Regression: Actual vs Predicted", x = "Actual", y = "Prec
theme_minimal()
# 2. Logistic Regression
log_model <- glm(Outcome ~ ., data = train_data, family = "binomial")
log_prob <- predict(log_model, test_data, type = "response")
log_class <- as.factor(ifelse(log_prob > 0.5, 1, 0))
cm_log <- confusionMatrix(log_class, test_data$Outcome)
accuracy_log <- cm_log$overall["Accuracy"]
accuracy_results <- rbind(accuracy_results, data.frame(Model = "Logistic Regress
# Plot: Predicted probabilities by class
ggplot(data.frame(Actual = test_data$Outcome, Predicted_Prob = log_prob), aes(x
geom_boxplot(aes(fill = Actual)) +
labs(title = "Logistic Regression: Predicted Probabilities", x = "Actual Outcc
theme_minimal()
# 3. Naive Bayes
nb_model <- naiveBayes(Outcome ~ ., data = train_data)
nb_predictions <- predict(nb_model, test_data)
cm_nb <- confusionMatrix(nb_predictions, test_data$Outcome)
accuracy_nb <- cm_nb$overall["Accuracy"]
56:1 (Top Level) R Script
Environment History Connections Tutorial
Console Background Jobs
R 4.4.1 · C:/Users/risha/OneDrive/Desktop/Predictive Project/
> log_class <- as.factor(ifelse(log_prob > 0.5, 1, 0))
> cm_log <- confusionMatrix(log_class, test_data$Outcome)
> accuracy_log <- cm_log$overall["Accuracy"]
> accuracy_results <- rbind(accuracy_results, data.frame(Model =
"Logistic Regression", Accuracy = accuracy_log))
> # Plot: Predicted probabilities by class
> ggplot(data.frame(Actual = test_data$Outcome, Predicted_Prob = l
og_prob), aes(x = Actual, y = Predicted_Prob)) +
+ geom_boxplot(aes(fill = Actual)) +
+ labs(title = "Logistic Regression: Predicted Probabilities", x
= "Actual Outcome", y = "Predicted Probability") +
+ theme_minimal()
Files Plots Packages Help Viewer Presentation
Zoom Export Publish
Logistic Regression: Predicted Probabilities
Predicted Probability
1.00
0.75
0.50
0.25
0.00
0
1
Actual Outcome
Actual
0
1


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1* ProjectLR*
Source on Save Run Source
Plot: Actual vs Predicted
ggplot(data.frame(Actual = as.numeric(test_data$Outcome), Predicted = lm_predict
geom_point(alpha = 0.5, color = "blue") +
geom_smooth(method = "lm", color = "red") +
labs(title = "Linear Regression: Actual vs Predicted", x = "Actual", y = "Prec
theme_minimal()
# 2. Logistic Regression
log_model <- glm(Outcome ~ ., data = train_data, family = "binomial")
log_prob <- predict(log_model, test_data, type = "response")
log_class <- as.factor(ifelse(log_prob > 0.5, 1, 0))
cm_log <- confusionMatrix(log_class, test_data$Outcome)
accuracy_log <- cm_log$overall["Accuracy"]
accuracy_results <- rbind(accuracy_results, data.frame(Model = "Logistic Regress
# Plot: Predicted probabilities by class
ggplot(data.frame(Actual = test_data$Outcome, Predicted_Prob = log_prob), aes(x
geom_boxplot(aes(fill = Actual)) +
labs(title = "Logistic Regression: Predicted Probabilities", x = "Actual Outcc
theme_minimal()
# 3. Naive Bayes
nb_model <- naiveBayes(Outcome ~ ., data = train_data)
nb_predictions <- predict(nb_model, test_data)
cm_nb <- confusionMatrix(nb_predictions, test_data$Outcome)
accuracy_nb <- cm_nb$overall["Accuracy"]
accuracy_results <- rbind(accuracy_results, data.frame(Model = "Naive Bayes", Ac
# Plot: Predicted probabilities by class
nb_prob <- predict(nb_model, test_data, type = "raw")
nb_prob_df <- data.frame(Actual = test_data$Outcome, Prob_Class_0 = nb_prob[, 1]
ggplot(nb_prob_df, aes(x = Actual)) +
geom_boxplot(aes(y = Prob_Class_0, fill = "Class 0"), alpha = 0.5) +
geom_boxplot(aes(y = Prob_Class_1, fill = "Class 1"), alpha = 0.5) +
labs(title = "Naive Bayes: Predicted Probabilities", x = "Actual Outcome", y =
6:17 (Top Level) R Script
Environment History Connections Tutorial
Console Background Jobs
R 4.4.1 · C:/Users/risha/OneDrive/Desktop/Predictive Project/
> accuracy_lm <- cm_lm$overall["Accuracy"]
> accuracy_results <- rbind(accuracy_results, data.frame(Model =
"Linear Regression", Accuracy = accuracy_lm))
> # Plot: Actual vs Predicted
> ggplot(data.frame(Actual = as.numeric(test_data$Outcome), Predic
ted = lm_predictions), aes(x = Actual, y = Predicted)) +
+ geom_point(alpha = 0.5, color = "blue") +
+ geom_smooth(method = "lm", color = "red") +
+ labs(title = "Linear Regression: Actual vs Predicted", x = "Ac
tual", y = "Predicted") +
+ theme_minimal()
`geom_smooth()` using formula = 'y ~ x'
Files Plots Packages Help Viewer Presentation
Zoom Export Publish
Linear Regression: Actual vs Predicted
Predicted
2.0
1.5
1.0
0.5
1.00
1.25
1.50
1.75
2.00
Actual



7. List of Analysis with Results

After analyzing the performance of each model, the Decision Tree model was found to yield the highest accuracy, making it the best choice for this dataset. This finding underscores the Decision Tree's capability to handle binary classification in a structured, interpretable way.

8. Future Scope

Future work on this project could include:

1. **Expanding the Dataset**: Incorporating additional health metrics could provide more comprehensive insights.
2. **Ensemble Modeling**: Using ensemble techniques like Random Forests or Gradient Boosting to potentially improve accuracy.
3. **Implementing Model Tuning**: Applying hyperparameter tuning to optimize the performance of each algorithm.

This could enhance the predictive power and reliability of the models, enabling further application in clinical environments.