Classification Modelling of infectious diseases

Breast Cancer- Classification

This breast cancer domain was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. This is one of three domains provided by the Oncology Institute that has repeatedly appeared in the machine learning literature. (See also lymphography and primary-tumor.)

Objectives

The data set is freely available at the **UC Irvine Machine Learning Repository. For this data set, t**he repository provides a comparison of several machine learning classification algorithms (XGBoost, Random Forest, Neural Networks, Support Vector Machine (SVM) and Logistic regression) through a graph of accuracy and precision metric implemented in Python.

Here, we delve into a greater details of the data set, the variables, and the implementation and comparison of all the above algorithms in R using tidymodels which is is a collection of packages for modeling and machine learning using tidyverse principles.

Data

The dataset consists of the following features with categorical and integer data types:

1. **Class**: Target variable with binary categories (no-recurrence-events, recurrence-events)
2. **Age**: Categorical, representing age groups (10-19, 20-29, etc.)
3. **Menopause**: Categorical, representing menopausal status (lt40, ge40, premeno).
4. **Tumor-size**: Categorical, representing tumor size ranges (0-4, 5-9, etc.)
5. **Inv-nodes**: Categorical, representing involved nodes (0-2, 3-5, etc.).

6. **Node-caps**: Binary categorical (yes, no).
7. **Deg-malig**: Integer, representing the degree of malignancy (1, 2, 3).
8. **Breast**: Binary categorical (left, right).
9. **Breast-squad**: Categorical, representing the quadrant of the breast (left-up, left-low, etc.).
10. **Irradiat**: Binary categorical (yes, no).

Let's start it here

Install packages

```
# Install necessary packages if you haven't already

#install.packages(c("tidymodels", "xgboost", "kernlab", #"randomForest", "nnet", "readxl", "

# Load the necessary libraries
library(tidymodels)
library(xgboost)
library(kernlab)
library(randomForest)
library(nnet)
library(ggplot2)
library(readxl)
library(tidyverse)
library(recipes)
```

Import data

```
# Import the dataset
data <- read_excel("DataSets/BreastCancer_Classification/BreastCancer_1988_Classification.xls

# View the first few rows of the dataset to understand its structure
glimpse(data)
```

```
Rows: 286
Columns: 10
$ Class          <chr> "no-recurrence-events", "no-recurrence-events", "no-rec~
```

```
$ Age            <chr> "30-39", "40-49", "40-49", "60-69", "40-49", "60-69", "~
$ menopause      <chr> "premeno", "premeno", "premeno", "ge40", "premeno", "ge~
$ `tumor-size`   <chr> "30-34", "20-24", "20-24", "15-19", "0-4", "15-19", "25~
$ `inv-nodes`    <chr> "0-2", "0-2", "0-2", "0-2", "0-2", "0-2", "0-2", "0-2",~
$ `node-caps`    <chr> "no", "no", "no", "no", "no", "no", "no", "no", "no", "~
$ `deg-malig`    <dbl> 3, 2, 2, 2, 2, 2, 2, 1, 2, 2, 3, 2, 1, 3, 3, 1, 2, 3, 3~
$ breast         <chr> "left", "right", "left", "right", "right", "left", "lef~
$ `breast-squad` <chr> "left_low", "right_up", "left_low", "left_up", "right_l~
$ irradiat       <chr> "no", "no", "no", "no", "no", "no", "no", "no", "no", "~
```

Data Wrangling Steps

1. **Convert Binary Text Features to Binary Numeric**:

- Convert node-caps, breast, and irradiat into binary numeric form (e.g., yes/no to 1/0).

2. **Encode Categorical Variables**:

- Convert categorical variables (Age, menopause, tumor-size, inv-nodes, breast-squad) into
- For models that can handle categorical data (e.g., Random Forest, Logistic Regression), e

3. **Ensure Numeric Representation**:

- Ensure all numeric features (e.g., deg-malig) are in numeric format.

The handling of features like tumor-size, menopause, Age, breast, and breast-quad by different algorithms in R depends on whether the algorithm can natively handle categorical data or if it requires numeric input. Here's how these features are typically processed:

1. Tumor-Size, Menopause, Age, Breast-Quad (Categorical Features)

- **Categorical features** like tumor-size, menopause, Age, and breast-quad are initially i

Handling by Different Algorithms:

- **XGBoost**:

- XGBoost cannot handle categorical data directly, so these features need to be **one-hot e

- **Support Vector Machine (SVM)**:

- Like XGBoost, SVM requires numeric inputs. Hence, these features should be **one-hot enc
- **Random Forest**:
- Random Forest can handle categorical features if they are provided as factors in R. Howe
- **Neural Networks**:
- Neural Networks typically expect numeric input, so these features should be **one-hot en
- **Logistic Regression**:
- Logistic Regression requires numeric input. If you use a library like glm, you can handl

2. Breast (Binary Feature)

- **Breast** is already a binary feature (left or right), but it needs to be converted int

Handling by Different Algorithms:

- **XGBoost, SVM, Random Forest, Neural Networks, Logistic Regression**:
- All these algorithms will work with breast after it has been converted to numeric form (

Summary of Transformation:

- **One-Hot Encoding**: Required for categorical variables like tumor-size, menopause, Age
- **Binary Encoding**: Convert binary features like breast from text (left, right) to numer
- **Factors**: For Random Forest, categorical features can remain as factors in R, but one

```r
# Apply label encoding to the target variable 'Class'
data <- data %>%
  mutate(Class = ifelse(Class == "no-recurrence-events", 0, 1))%>%
  mutate(Class=as.factor(Class))

# One-Hot Encoding for categorical variables
data <- recipe(Class ~ ., data = data) %>%
  step_dummy(all_nominal_predictors(), -all_outcomes()) %>%
```

```
  prep() %>%
  bake(new_data = NULL)

# Ensure all integer features are in the correct format
data <- data %>%
  mutate(`deg-malig` = as.integer(`deg-malig`))

# View the transformed data
head(data)
```

```
# A tibble: 6 x 34
  `deg-malig` Class Age_X30.39 Age_X40.49 Age_X50.59 Age_X60.69 Age_X70.79
        <int> <fct>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
1           3 0              1          0          0          0          0
2           2 0              0          1          0          0          0
3           2 0              0          1          0          0          0
4           2 0              0          0          0          1          0
5           2 0              0          1          0          0          0
6           2 0              0          0          0          1          0
# i 27 more variables: menopause_lt40 <dbl>, menopause_premeno <dbl>,
#   `tumor-size_X10.14` <dbl>, `tumor-size_X15.19` <dbl>,
#   `tumor-size_X20.24` <dbl>, `tumor-size_X25.29` <dbl>,
#   `tumor-size_X30.34` <dbl>, `tumor-size_X35.39` <dbl>,
#   `tumor-size_X40.44` <dbl>, `tumor-size_X45.49` <dbl>,
#   `tumor-size_X5.9` <dbl>, `tumor-size_X50.54` <dbl>,
#   `inv-nodes_X12.14` <dbl>, `inv-nodes_X15.17` <dbl>, ...
```

3. Split the Data into Training and Testing Sets

```
# Split data into training and testing sets
set.seed(123)
data_split <- initial_split(data, prop = 0.75)
train_data <- training(data_split)
test_data <- testing(data_split)
```

4. Create a Recipe for Data Preprocessing

```
# Create a recipe for preprocessing
data_recipe <- recipe(Class ~ ., data = train_data) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors()) %>% # remove zero variance predictors
  step_normalize(all_predictors())
```

5. Define Models

```
# Logistic Regression
log_reg_model <- logistic_reg() %>%
  set_engine("glm")

# Random Forest
rf_model <- rand_forest(mtry = 3, trees = 1000) %>%
  set_engine("randomForest") %>%
  set_mode("classification")

# Support Vector Machine
svm_model <- svm_rbf(cost = 1) %>%
  set_engine("kernlab") %>%
  set_mode("classification")

# XGBoost
xgb_model <- boost_tree(trees = 1000, tree_depth = 3, min_n = 2) %>%
  set_engine("xgboost") %>%
  set_mode("classification")

# Neural Network
nn_model <- mlp(hidden_units = 5, penalty = 0.1, epochs = 100) %>%
  set_engine("nnet") %>%
  set_mode("classification")
```

6. Create Workflows

```
# Create workflows for each model
log_reg_workflow <- workflow() %>%
  add_model(log_reg_model) %>%
  add_recipe(data_recipe)

rf_workflow <- workflow() %>%
  add_model(rf_model) %>%
  add_recipe(data_recipe)

svm_workflow <- workflow() %>%
  add_model(svm_model) %>%
  add_recipe(data_recipe)

xgb_workflow <- workflow() %>%
```

```
  add_model(xgb_model) %>%
  add_recipe(data_recipe)

nn_workflow <- workflow() %>%
  add_model(nn_model) %>%
  add_recipe(data_recipe)
```

7. Fit the Models

```
# Fit the models
log_reg_fit <- fit(log_reg_workflow, data = train_data)
rf_fit <- fit(rf_workflow, data = train_data)
svm_fit <- fit(svm_workflow, data = train_data)
xgb_fit <- fit(xgb_workflow, data = train_data)
nn_fit <- fit(nn_workflow, data = train_data)
```

8. Evaluate the Models

```
# Evaluate the models on the test set
log_reg_results <- predict(log_reg_fit, test_data) %>%
  bind_cols(test_data) %>%
  metrics(truth = Class, estimate = .pred_class)

rf_results <- predict(rf_fit, test_data) %>%
  bind_cols(test_data) %>%
  metrics(truth = Class, estimate = .pred_class)

svm_results <- predict(svm_fit, test_data) %>%
  bind_cols(test_data) %>%
  metrics(truth = Class, estimate = .pred_class)

xgb_results <- predict(xgb_fit, test_data) %>%
  bind_cols(test_data) %>%
  metrics(truth = Class, estimate = .pred_class)

nn_results <- predict(nn_fit, test_data) %>%
  bind_cols(test_data) %>%
  metrics(truth = Class, estimate = .pred_class)
```

9. Collect the Results for Comparison

```r
# Collect all metrics
model_results <- bind_rows(
  log_reg_results %>% mutate(Model = "Logistic Regression"),
  rf_results %>% mutate(Model = "Random Forest"),
  svm_results %>% mutate(Model = "SVM"),
  xgb_results %>% mutate(Model = "XGBoost"),
  nn_results %>% mutate(Model = "Neural Network")
)

# Filter for Accuracy and Precision metrics
model_results_filtered <- model_results %>%
  filter(.metric %in% c("accuracy", "precision"))

# Summarize the mean and variation
model_summary <- model_results_filtered %>%
  group_by(Model, .metric) %>%
  summarize(
    mean_value = mean(.estimate),
    sd_value = sd(.estimate),
    .groups = "drop"
  )
```

7. Visualize the results

```r
# Create bar graphs with mean performance and variations
ggplot(model_summary, aes(x = Model, y = mean_value, fill = .metric)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  geom_errorbar(aes(ymin = mean_value - sd_value, ymax = mean_value + sd_value),
                width = 0.2, position = position_dodge(.9)) +
  geom_point(position = position_dodge(.9)) +
  labs(title = "Model Performance Comparison",
       x = "Models",
       y = "Metric Scores",
       fill = "Metrics") +
  theme_minimal() +
  coord_flip()
```

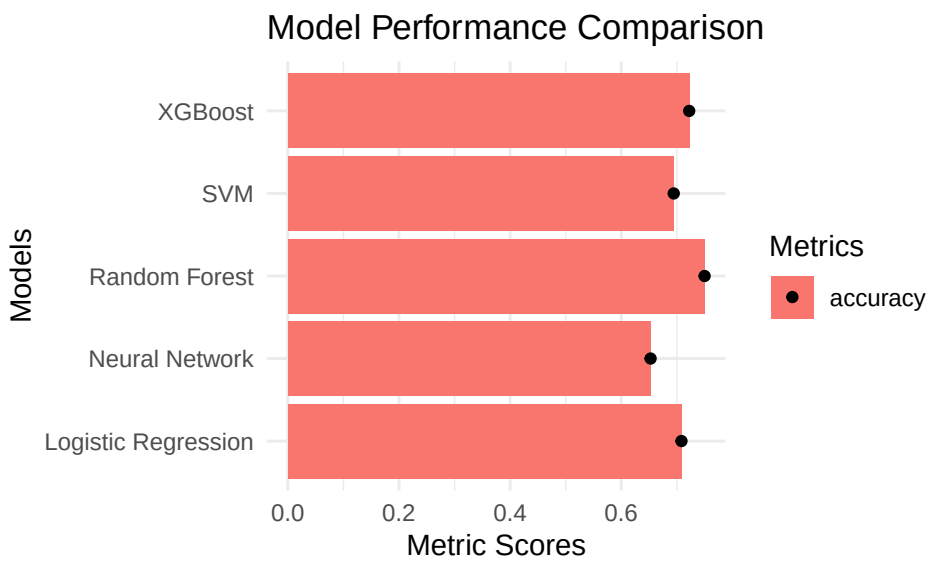


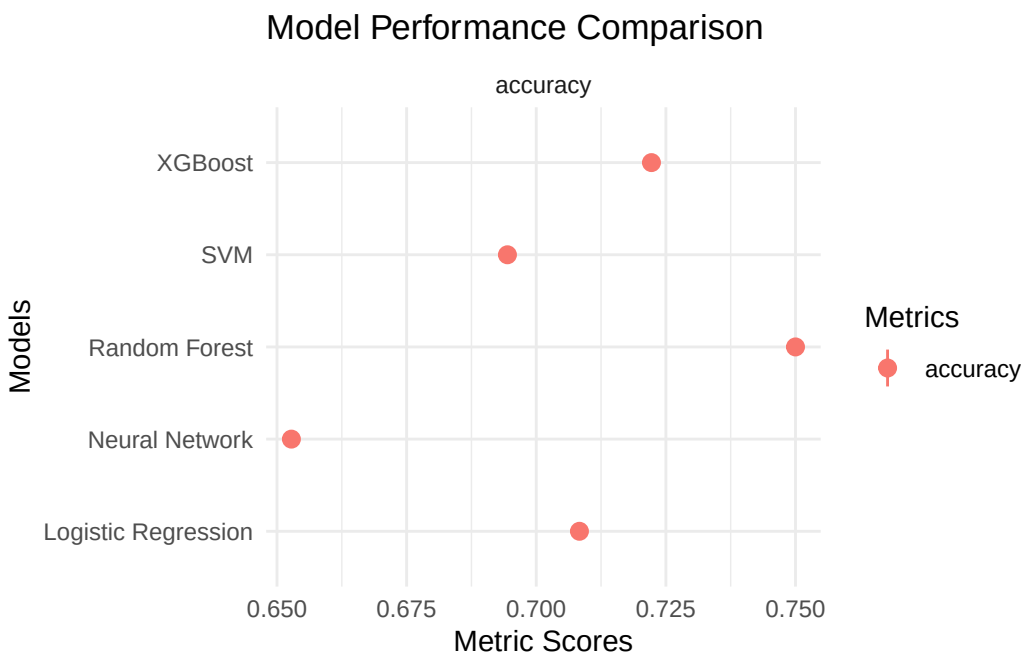
Figure 1: A scatter plot

```
library(ggplot2)

# Assume 'model_summary' is your data frame that contains the following columns:
# - Model: The name of the model (e.g., "XGBoost", "SVM", etc.)
# - mean_value: The mean score of the metric (accuracy or precision)
# - sd_value: The standard deviation of the score
# - .metric: The name of the metric ("accuracy" or "precision")

# Create a two-panel plot using point range plots
ggplot(model_summary, aes(x = Model, y = mean_value, color = .metric)) +
  geom_pointrange(aes(ymin = mean_value - sd_value, ymax = mean_value + sd_value),
                  position = position_dodge(0.5)) +
  facet_wrap(~ .metric, scales = "free_y") +
  labs(title = "Model Performance Comparison",
       x = "Models",
       y = "Metric Scores",
       color = "Metrics") +
  theme_minimal() +
  coord_flip()
```

```
Warning: Removed 5 rows containing missing values or values outside the scale range
(`geom_segment()`).
```



References

(Kuhn and Wickham 2020)

Rivers et al. (2019)

Kuhn, Max, and Hadley Wickham. 2020. "Tidymodels: A Collection of Packages for Modeling and Machine Learning Using Tidyverse Principles." https://www.tidymodels.org.

Rivers, Caitlin, Jean-Paul Chretien, Steven Riley, Julie A Pavlin, Alexandra Woodward, David Brett-Major, Irina Maljkovic Berry, Lindsay Morton, Richard G Jarman, and Matthew Biggerstaff. 2019. "Using "Outbreak Science" to Strengthen the Use of Models During Epidemics." *Nature Communications* 10 (1): 3102.