Making optimal use of ChatGPT and other chatbots for data science

Ronald (Ryy) Glenn Thomas

2024-09-07

Table of contents

1 Introduction

" ChatGPT stunned the world on its launch in November 2022. Powered by a large language model (LLM) and trained on much of the text published on the Internet, the artificial intelligence (AI) chatbot, created by OpenAI in San Francisco, California, makes the latest advances in natural-language processing broadly accessible by providing a dialogue-based interface capable of answering complex questions, composing sophisticated essays and generating source code. One obvious question was: how could this tool improve science? "

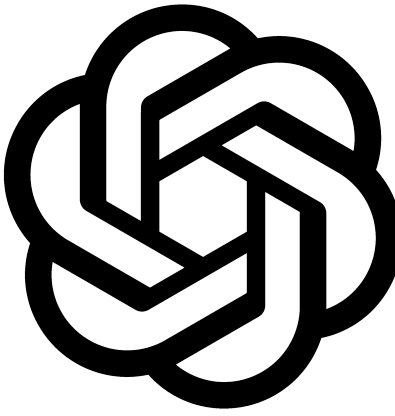

Figure 1: quarto

2 Example work up of a regression analysis for the iris data set.

1. Prompt: "I'm doing a logistic regression analysis and want to K-fold cross-validation"
2. Prompt: "I want to do the analysis in R"
3. Prompt: "I want to use an ROC curve to visualize the results"
4. Prompt: "list all the R analysis code in one block"
5. Prompt: "list all my R analysis related prompts for just now in one block"
6. Prompt: "should I split my data set into train and test parts if I'm using k-fold cross-validation?"
7. Prompt: "extend the code for the R logistic regression analysis to split the iris data into test and training sets before applying k-fold cross-validation"
8. Prompt: "add a 95% confidence interval calculation to the AUC annotation in the ROC plot"
9. Prompt: "do the same analysis but use lasso regression instead of logistic regression"

3 R code

```r
install.packages("caret")  # Install if not already installed
install.packages("pROC")   # Install if not already installed
install.packages("glmnet") # Install if not already installed
library(caret)
library(pROC)
library(glmnet)

# Prepare binary dataset
data(iris)
iris_binary <- iris[iris$Species != "setosa", ]
iris_binary$Species <- factor(iris_binary$Species)
```

```r
# Split the data into training and test sets
set.seed(42)
trainIndex <- createDataPartition(iris_binary$Species, p = 0.8, list = FALSE)
train_data <- iris_binary[trainIndex, ]
test_data <- iris_binary[-trainIndex, ]

# Convert the data into matrix format as required by glmnet
x_train <- model.matrix(Species ~ ., train_data)[,-1]
y_train <- train_data$Species

x_test <- model.matrix(Species ~ ., test_data)[,-1]
y_test <- test_data$Species

# Set up cross-validation for Lasso regression on the training data
train_control <- trainControl(method = "cv", number = 5, classProbs = TRUE, summaryFunction = t

# Define the lasso model using glmnet (alpha=1 for Lasso)
lasso_model <- train(x = x_train, y = y_train, method = "glmnet", trControl = train_control, tu

# Make predictions on the test set
test_predictions <- predict(lasso_model, newdata = x_test, type = "prob")[,2]

# Plot ROC curve for the test set
roc_curve_test <- roc(y_test, test_predictions, levels = rev(levels(y_test)))
plot(roc_curve_test, col = "blue", lwd = 2, main = "ROC Curve for Lasso Regression on Test Data

# Calculate AUC and its 95% confidence interval
auc_value <- auc(roc_curve_test)
ci <- ci.auc(roc_curve_test, conf.level = 0.95)

# Add AUC with 95% confidence interval to the plot
legend_text <- paste("AUC =", round(auc_value, 2), "\n95% CI:", round(ci[1], 2), "-", round(ci
legend("bottomright", legend = legend_text, col = "blue", lwd = 2)
```

4 Complete R Code for Lasso Regression with LOOCV on the Entire Dataset

```r
# Prepare binary dataset
data(iris)
iris_binary <- iris[iris$Species != "setosa", ]
iris_binary$Species <- factor(iris_binary$Species)

# Convert the data into matrix format as required by glmnet
x <- model.matrix(Species ~ ., iris_binary)[,-1]
y <- iris_binary$Species

# Set up LOOCV for Lasso regression on the entire dataset
train_control <- trainControl(method = "LOOCV", classProbs = TRUE, summaryFunction = twoClassSu

# Define the lasso model using glmnet (alpha=1 for Lasso)
lasso_model <- train(x = x, y = y, method = "glmnet", trControl = train_control, tuneGrid = exp

# Make predictions using LOOCV
predictions <- predict(lasso_model, newdata = x, type = "prob")[,2]

# Plot ROC curve for the entire dataset
roc_curve <- roc(y, predictions, levels = rev(levels(y)))
plot(roc_curve, col = "blue", lwd = 2, main = "ROC Curve for Lasso Regression with LOOCV")

# Calculate AUC and its 95% confidence interval
auc_value <- auc(roc_curve)
ci <- ci.auc(roc_curve, conf.level = 0.95)

# Add AUC with 95% confidence interval to the plot
legend_text <- paste("AUC =", round(auc_value, 2), "\n95% CI:", round(ci[1], 2), "-", round(ci
legend("bottomright", legend = legend_text, col = "blue", lwd = 2)
```