Module 13 Challenge

Start Assignment

Due Dec 19, 2024 by 11:59pm Points 100
Submitting a text entry box or a website url Attempts 0
Allowed Attempts 4

Submission
✕ Not Submitted!
Submission Details
Grade: 0 (100 pts possible)
Graded Anonymously: no
Comments:
No Comments

Background

Let's say you work at an Internet Service Provider (ISP) and you've been tasked with improving the email filtering system for its customers. You've been provided with a dataset that contains information about emails, with two possible classifications: spam and not spam. The ISP wants you to take this dataset and develop a supervised machine learning (ML) model that will accurately detect spam emails so it can filter them out of its customers' inboxes.

You will be creating two classification models to fit the provided data, and evaluate which model is more accurate at detecting spam. The models you'll create will be a logistic regression model and a random forest model.

Files

Download the following files to help you get started:

Module 13 Challenge files

Before You Begin

Before starting the assignment, be sure to complete the following steps:

- Create a new repository for this project called `classification-challenge`. **Do not add this homework assignment to an existing repository**.
- Clone the new repository to your computer.
- Inside your local Git repository, add the starter file `spam_detector.ipynb` from your downloads.
- Push these changes to GitHub or GitLab.

Instructions

This challenge consists of the following subsections:

- Split the data into training and testing sets.
- Scale the features.
- Create a logistic regression model.
- Create a random forest model.
- Evaluate the models.

Split the Data into Training and Testing Sets

Open the starter code notebook and then use it to complete the following steps.

1. Read the data from https://static.bc-edx.com/ai/ail-v-1-0/m13/challenge/spam-data.csv into a Pandas DataFrame.
2. In the appropriate markdown cell, make a prediction as to which model you expect to do better.
3. Create the labels set (y) from the "spam" column, and then create the features (X) DataFrame from the remaining columns.

   NOTE

   A value of 0 in the "spam" column means that the message is legitimate. A value of 1 means that the message has been classified as spam.

4. Check the balance of the labels variable (y) by using the `value_counts` function.
5. Split the data into training and testing datasets by using `train_test_split`.

Scale the Features

1. Create an instance of `StandardScaler`.
2. Fit the Standard Scaler with the training data.
3. Scale the training and testing features DataFrames using the transform function.

Create a Logistic Regression Model

Employ your knowledge of logistic regression to complete the following steps:

1. Fit a logistic regression model by using the scaled training data (`X_train_scaled` and `y_train`). Set the `random_state` argument to `1`.
2. Save the predictions on the testing data labels by using the testing feature data (`X_test_scaled`) and the fitted model.
3. Evaluate the model's performance by calculating the accuracy score of the model.

Create a Random Forest Model

Employ your knowledge of the random forest classifier to complete the following steps:

1. Fit a random forest classifier model by using the scaled training data (`X_train_scaled` and `y_train`).
2. Save the predictions on the testing data labels by using the testing feature data (`X_test_scaled`) and the fitted model.
3. Evaluate the model's performance by calculating the accuracy score of the model.

Evaluate the Models

In the appropriate markdown cell, answer the following questions:

1. Which model performed better?
2. How does that compare to your prediction?

Requirements

To receive all points, your Jupyter notebook file must have all of the following:

Split the Data into Training and Testing Sets (30 points)

- There is a prediction about which model you expect to do better. (5 points)
- The labels set (y) is created from the "spam" column. (5 points)
- The features DataFrame (X) is created from the remaining columns. (5 points)
- The `value_counts` function is used to check the balance of the labels variable (y). (5 points)
- The data is correctly split into training and testing datasets by using `train_test_split`. (10 points)

Scale the Features (20 points)

- An instance of `StandardScaler` is created. (5 points)
- The Standard Scaler instance is fit with the training data. (5 points)
- The training features DataFrame is scaled using the transform function. (5 points)
- The testing features DataFrame is scaled using the transform function. (5 points)

Create a Logistic Regression Model (20 points)

- A logistic regression model is created with a `random_state` of `1`. (5 points)
- The logistic regression model is fitted to the scaled training data (`X_train_scaled` and `y_train`). (5 points)
- Predictions are made for the testing data labels by using the testing feature data (`X_test_scaled`) and the fitted model, and saved to a variable. (5 points)
- The model's performance is evaluated by calculating the accuracy score of the model with the `accuracy_score` function. (5 points)

Create a Random Forest Model (20 points)

- A random forest model is created with a `random_state` of `1`. (5 points)
- The random forest model is fitted to the scaled training data (`X_train_scaled` and `y_train`). (5 points)
- Predictions are made for the testing data labels by using the testing feature data (`X_test_scaled`) and the fitted model, and saved to a variable. (5 points)
- The model's performance is evaluated by calculating the accuracy score of the model with the `accuracy_score` function. (5 points)

Evaluate the Models (10 points)

The following questions are answered accurately:

- Which model performed better? (5 points)
- How does that compare to your prediction? (5 points)

Grading

This challenge will be evaluated against the requirements and assigned a grade according to the following table:

Grade	Points
A (+/-)	90+
B (+/-)	80–89
C (+/-)	70–79
D (+/-)	60–69
F (+/-)	< 60

Submission

To submit your Challenge assignment, click Submit, and then provide the URL of your GitHub repository for grading.

NOTE

You are allowed to miss up to two Challenge assignments and still earn your certificate. If you complete all Challenge assignments, your lowest two grades will be dropped. If you wish to skip this assignment, click Next, and move on to the next module.

Comments are disabled for graded submissions in Bootcamp Spot. If you have questions about your feedback, please notify your instructional staff or your Student Success Manager. If you would like to resubmit your work for an additional review, you can use the Resubmit Assignment button to upload new links. You may resubmit up to three times for a total of four submissions.

IMPORTANT

It is your responsibility to include a note in the README section of your repo specifying code source and its location within your repo. This applies if you have worked with a peer on an assignment, used code that you did not author or create, source code from a forum such as Stack Overflow, or received code outside curriculum content from support staff, such as an Instructor, TA, Tutor, or Learning Assistant. This will provide visibility to grading staff of your circumstance in order to avoid flagging your work as plagiarized.

If you are struggling with a challenge assignment or any aspect of the academic curriculum, please remember that there are student support services available for you:

1. Ask the class Slack channel/peer support.
2. AskBCS Learning Assistants exists in your class Slack application.
3. Office hours facilitated by your instructional staff before and after each class session.
4. Tutoring Guidelines - schedule a tutor session in the Tutor Sessions section of Bootcampspot - Canvas
5. If the above resources are not applicable and you have a need, please reach out to a member of your instructional team, your Student Success Advisor, or submit a support ticket in the Student Support section of your BCS application.

Reference

Hopkins, M., Reeber, E., Forman, G. & Suermondt, J. 1999. Spambase [Dataset]. UCI Machine Learning Repository. Available: https://archive.ics.uci.edu/dataset/94/spambase [2023, April 28].