Week_3_exercise

October 30, 2023

1 Week 2: Feature Attribution: Shapley Value and SHAP

Author: Van Bach Nguyen, licensed under the Creative Commons Attribution 3.0 Unported License https://creativecommons.org/licenses/by/3.0/
Based on: https://github.com/marcotcr/lime

2 Table of contents

- Exercise 1: Shapley Value (3 Points)
- Exercise 2: SHAP (3 Points)

3 Exercise 1: Shapley value calculation (3 Points + 1 bonus point)

Description: In this exercise, we will practice calculating Shapley values using Python.

Goal: The goal of this exercise is to gain a deep understanding of Shapley Value calculation and learn how to implement it.

Task: Implement question 1 from the Theoretical Exercise in the exercise sheet to verify your calculation results.

Note: - Your results, interpretation, and comments on the results are more important for evaluating the exercises than your code. - The bonus point is an extra point.

Exercise 1.1: Grade function (1 Point) The *grade* function should take a group of students as input and return the grade for this group.
Input: a group of students
Output: The grade for the group of students.
For example: *grade([A, B, C])* should return 10.

Task: Implement the *grade* function with the description below

[157]:

```python
#Implement the function grade with the description below
def grade(students):
    """
    Calculate the grade for a group of collaborated students.

    Args:
        students (list): A list of student names who collaborated on the␣
↪assignment.

    Returns:
        int: The calculated grade for the group.

    The function takes a list of student names who collaborated on a project␣
↪and calculates
    the group's grade based on their performance and contributions.
    """
```

Task: Print the grades for all 8 possible groups of students. They should correspond to the exercise's description.

[ ]:

```
#Your code
```

Exercise 1.2: Implement the Shapley value calculation (2 Points) The *calculate_shapley* function should take the *grade* function and a student (e.g "A") as inputs and return the Shapley value for this student with respect to the *grade* function.
Input: - *func:* The grade function used for calculating the Shapley value.
- *Student:* The student for whom the Shapley value needs to be calculated.

Output: Shapley value of the student.
For example: *calculate_shapley(grade, "A")* = V where V is the Shapley value of student "A" with respect to the provided *func* grade function.

Task: Implement the function *calculate_shapley* with the description below

[158]:

```
#Implement the function with the description below
def calculate_shapley(func=grade, student="A"):
    """
    Calculate the Shapley value of a student using a given grade function.

    Args:
        func (function): The grade function that returns the grade for a group␣
↪of students.
        student (str): The student for whom the Shapley value is to be␣
↪calculated (e.g., "A", "B", or "C").

    Returns:
        float: The Shapley value of the specified student.

    This function takes a grade function and a student as inputs and calculates␣
↪the Shapley value
    for the given student based on the provided grade function.
    """
```

Task: Use the implemented *calculate_shapley* function to calculate the Shapley value for each student (A, B, C) and print the results.

[ ]:

```
#Your code
```

Task: Compare the results from the calculate_shapley function with your manual calculations in Theoretical Exercise Question 1. The results should match; if they do not, please review your calculations/implementations.

3.0.1 Exercise 1.2 (1 bonus point)

Each student represents a feature, and the function represents a model. Extend the *calculate_shapley* function to take a (sklearn) machine learning model trained on a dataset, a feature, and a dataset as inputs, and calculate the Shapley value for the specified feature.

[ ]:

```
#your own implementation
def calculate_shapley_extended(model, dataset, feature):
    """
    Calculate the Shapley value of a feature given a model trained on a dataset.

    Args:
        model: A sklearn machine learning model.
        dataset: The dataset on which the model was trained.
        feature (str): The feature in the dataset for which the Shapley value␣
 ↪is to be calculated.

    Returns:
        float: The Shapley value of the specified feature.
    """
```

4 Exercise 2: SHAP (3 Points)

Description: In this exercise, we will practice using the SHAP library.

Goal: To get familiar with SHAP library and Kernel SHAP

Task: Use SHAP to explain a model's prediction.

Note: Your results, interpretation, and comments on the results are more important for evaluating the exercises than your code.

Install SHAP by executing the command below:

[ ]:

```
# Run the command below once:
!pip install shap
```

The code below trains a random forest model on the Diabetes dataset:

[127]:

```
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
from sklearn.model_selection import train_test_split
df = pd.read_csv("dataset/diabetes.csv")
X = df.drop('Outcome', axis=1)
```

```
y = df['Outcome']
#Split data into train/test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
#Train a RF model
clf = RandomForestClassifier(max_depth=3, random_state=0)
clf = clf.fit(X_train, y_train)
```

Predict an instance, e.g. instance 12th.

[169]:

```
instance_id = 12
print(f"the label is {clf.predict([X_test.iloc[instance_id]]).item()}")
```

```
the label is 0
```

Exercise 2.1 (2 Points) Why does the model give that prediction? We can use Kernel SHAP in the SHAP library to explain it.

Task: Given the trained random forest classifier *clf*, explain the prediction of the model for the 12th instance in X_test using Kernel SHAP in the SHAP library.
Note: You can use the force_plot function to visualize the SHAP values, and you should provide your interpretation of the plot.

[ ]:

```
# Your implementation
```

Exercise 2.2 (1 Point) SHAP claims to have three desirable properties. One of them is local accuracy $f(x) \approx g(x')$

Task: Verify the local accuracy property of this explanation. In other words, calculate the $f(x)$ and $g(x')$ and compare them.

[ ]:

```
# Your implementation
```