Final Project: Building and serving a machine learning model with Streamlit

Welcome to the final project of our course! The objective of this project is to build a simple linear regression model to predict target values based on input features and then create an interactive web application to serve these predictions. This will involve generating a synthetic dataset, training a regression model, evaluating its performance, and finally, deploying the model through a Streamlit application that allows users to make predictions and visualize the results.

Installing Required Libraries

To ensure a smooth development experience for your final project, all the necessary Python libraries and dependencies have been listed in a `requirements.txt` file.

To install the required libraries, follow these steps:
1. Open your terminal or command prompt.
2. Navigate to the directory containing your project files, including the `requirements.txt` file.
3. Run the following command:

```
pip install -r requirements.txt
```

This command tells **pip**, the Python package installer, to download and install all the libraries and their specific versions listed in the `requirements.txt` file.

Make sure you have Python and **pip** installed on your system before running this command. If you encounter any permission issues, you might need to add `sudo` at the beginning of the command (for macOS/Linux) or run the command prompt as an administrator (on Windows).

Part 1: Model development

In the first part of the project, you will focus on the machine learning aspects, including data preparation, model training, and serialization. The starter code in `model.py` guides you through the process of generating a synthetic dataset, training a linear regression model, evaluating its performance, and saving the trained model and datasets for future use.

Task breakdown

1. **Train a regression model:**
    - Implement the `train_regression_model` function. This function will take the training data (**X_train** and **y_train**) as input and return a trained linear regression model. Use the **LinearRegression** class from **sklearn.linear_model** for the model.
2. **Save the trained regression model:**
    - Implement the `save_regression_model` function. This function will serialize the trained regression model using the **dump** function from **joblib** and save it to a specified filename. The default filename should be **"linear_regression_model.joblib"**.
3. **Evaluate the regression model:**
    - Implement the `evaluate_regression_model` function. This function will take the trained model and test dataset (**X_test** and **y_test**) as input, predict the target values using the model, and then calculate and print the mean squared error (MSE) of the predictions.
4. **Serialize and save datasets:**
    - Implement the `save_initial_datasets` function. This function will serialize and save the entire feature matrix **X** and the corresponding target values **y** using **joblib**. The files should be saved as **"X.joblib"** and **"y.joblib"**, respectively.

Implementation Notes

- Pay attention to the documentation strings (docstrings) provided for each function. They offer guidance on the function's purpose, inputs, and expected outputs.

Part 2: Streamlit Web Application

The second part of the project shifts focus to deploying your trained model into a real-world application. Using the starter code in `app.py`, you will create a

Streamlit web application that enables users to input features and receive predictions from your model. This task will challenge you to implement functions for loading the model, making predictions, and visually comparing these predictions against actual values.

Task breakdown

1. **Implement `load_and_predict` function:**
    - This function is responsible for deserializing a saved regression model from a specified file and using this model to predict the target value for user-provided input data. Ensure that you handle the loading of the model correctly and perform prediction on the provided data.
2. **Set up the Streamlit Web application:**
    - Utilize the `create_streamlit_app` function to construct the Streamlit web app's user interface. This includes:
        - Displaying an app title.
        - Providing a slider for users to select or input a feature value for prediction.
        - Implementing a "Predict value" button that, when clicked, uses the `load_and_predict` function to generate and display a prediction and uses `visualize_difference` function display the difference between this model's prediction and the actual value on a chart.
3. **Implement `visualize_difference` function:**
    - This function aims to load the original datasets (features and targets), find the actual target value corresponding to the user-selected input feature, and calculate the difference between this actual value and the model's prediction. You will:
        - Load the original datasets.
        - Identify the actual target value that corresponds to the provided input feature.
        - Visualize the entire dataset, the actual target, and the predicted target on a scatter plot, highlighting the difference between the actual and predicted values. This involves plotting the dataset, marking the predicted and actual values with distinct colors, drawing a line to represent the difference, and annotating this difference on the plot.

Implementation Notes

- The visualization component requires knowledge of Matplotlib for generating scatter plots and annotations.
- While the skeleton code provides a structure, ensure to properly handle data types and formats, especially when interfacing between NumPy arrays and Streamlit UI components.
- Pay attention to the documentation strings (docstrings) provided for each function. They offer guidance on the function's purpose, inputs, and expected outputs.

Completion Criteria

Your final project will be considered complete when you have successfully implemented the tasks outlined in `model.py` and `app.py`, resulting in a functional web application that allows users to make predictions using the linear regression model you trained. This application should not only provide predictions but also facilitate an understanding of the model's performance through visualization. Here’s an example screenshot of how the application can look.

Simple Regression model prediction

Input Feature for Prediction

1.27

-3.00 3.00

Predict value

Prediction: 26.509899021941507