Capstone: The Art of Approximation 2023-2024

Digit Recognizer Project

Ojas Chaturvedi, Ritwik Jayaraman, Saianshul Vishnubhaktula, Zaheen Jamil

1 Proposal

1.1 Programming Language

Python, a simple and popular language for machine learning and data science due to its extensive libraries and frameworks

1.2 Objective

To develop a custom machine learning model which would be able to determine what a digit is from an image of a handwritten single digit

1.3 Implementation

1.3.1 Overview of steps

1. Data Exploration and Visualization
2. Data Preprocessing
3. Feature Engineering
4. Model Building
5. Model Training and Testing
6. Model Evaluation and Deployment
7. Hyperparameter Tuning and Optimization
8. Website/App Development

1.3.2 Potential Libraries

1. **Pandas:** For data manipulation and analysis
2. **NumPy:** For numerical computing and working with arrays
3. **Matplotlib:** For data visualization
4. **Scikit-learn:** For data mining and analysis
5. **TensorFlow:** For deep learning and complex neural network modeling
6. **Flask/Django:** For backend web development
7. **SQLAlchemy:** For SQL databases and Object-Relational Mapping

1.3.3 Manual Work

1. Making algorithms for data preprocessing and feature engineering
2. Building custom model
3. Training and testing model
4. Creating website/app that can use the model and store results for future training of model
5. Documentation of all steps

1.4 Jobs

1. **Machine/Deep Learning Developers**
   (a) Develops the machine learning model
   (b) Trains & tests the model
   (c) Makes the model usable in the website/app
2. **Data Analyst**
   (a) Algorithm development for preprocessing and feature engineering
   (b) Will still contribute as a Machine Learning Developer
3. **GUI Developer**

(a) Makes the website/app and all of its functionality (UI)
(b) Makes the model usable in the website/app
(c) Will still contribute as a Machine Learning Developer

2 Timeline

Note: Multiple drafts of the model will be created, with each draft having increased accuracy.

- Week 1:
  - Start learning Bayesian Statistics
    - Different types of Bayesian models in use
    - How different concepts (e.g. Gaussian) are used with Bayesian
- Week 2:
  - Continue Bayesian Statistics study
    - Explore types of Bayesian models for MNIST (for now we are thinking of using a Naive Bayesian model)
    - **Deadline:** Be finished learning what we need for Bayesian by Friday, December 22
- Week 3:
  - Implement initial machine learning models (make a first draft of the Bayesian model; this includes optimization techniques such as 5-fold cross-validation)
    - **Deadline:** Be finished with draft 1 by Friday, December 29
  - Begin exploratory data analysis on the MNIST dataset. Use Python to clean the dataset and prepare it for use to test ML model version 1
    - **Deadline:** Test model draft 1 by Friday, December 29
- Week 4:
  - Implement initial machine learning models (make a second draft of the Bayesian model)
    - **Deadline:** Be finished with draft 2 by Wednesday, January 3
    - **Deadline:** Test model draft 2 by Friday, January 5
  - Start developing the GUI that implements the model (initial framework)
    - **Deadline:** Decide GUI type (website or app) by Friday, January 5
- Week 5:

  - Implement initial machine learning models (make a third/nearly final draft of the Bayesian model)
    - **<u>Deadline:</u>** Be finished with draft 3 by Wednesday, January 10
    - **<u>Deadline:</u>** Test model draft 3 by Friday, January 12
  - Develop the image uploading/camera functionality
    - **<u>Deadline:</u>** Friday, January 12

- Week 6:
  - Refine the model based on testing of user-inputted data
    - Link GUI to model and use Python to clean image data (scaling resolution, grayscale) before inputting it into the test set
      - **<u>Deadline:</u>** Friday, January 19
  - Continue testing and model adjustments
  - Start class presentation
    - **<u>Deadline:</u>** Have the introduction + most of the methods slides finished by Friday, January 19

- Week 7:
  - Complete the development of the GUI
    - **<u>Deadline:</u>** Friday, January 26
  - Finalize the model after thorough testing and validation + complete final testing
    - **<u>Deadline:</u>** Be finished with all by Friday, January 26
  - Continue class presentation (specifically the methods + deliverable showcase slides)
    - **<u>Deadline:</u>** Finalized presentation by Friday, January 26

- Week 8:
  - Prepare and fine-tune presentation and documentation for the project submission
    - **<u>Deadline:</u>** Be finished by Tuesday, January 30

3 Libraries

To learn the essential Python libraries that were mentioned in the proposal (Pandas, NumPy, Matplotlib, Scikit-learn, TensorFlow, Flask/Django, SQLAlchemy), the best tool would be the official documentation for that library. We can also find YouTube tutorials about specific aspects of these libraries that are relevant to our project since we won't need to understand the entire library for our project.

4 Deliverables

Machine Learning Model: Trained model capable of recognizing and interpreting handwritten digits, using the MNIST dataset for training and validation. The model will incorporate Bayesian statistics for better accuracy (and, if possible, a well-defined cost function to enhance performance). The focus will be on creating a robust, efficient model that can accurately classify new, unseen handwritten digit images.
Graphical User Interface - Website or Application: This GUI will serve as the interactive front-end for the machine learning model. Some key features will be:

- Image Upload Capability: If designed as a web application, users can upload images of handwritten digits. If it's a standalone application, the application will be able to access the users' cameras to take a picture. These images could be stored using Firebase.
- Digit Recognition: Upon uploading an image, the model will analyze and display the predicted digit.
- User-Friendly Interface: The GUI will be designed to be intuitive and easy to navigate, ensuring accessibility for users with varying levels of technical expertise.
- Feedback Mechanism: An option for users to provide feedback on the model's predictions, which can be valuable for further model improvement.
- Cross-Platform Compatibility: If designed as a web application, it will be accessible across various devices and browsers. It could be hosted on GitHub for free to run 24/7. If it's a standalone application, it may be designed for different operating systems using a framework like Flutter or made in React for Android.