May 11, 2024

C964: Computer Science Capstone Template

Part A: Letter of Transmittal

05/11/2024

Mr. Monty, Head of Special Collections at NaN University

NaN University

1234 Placeholder Blvd, Null PA

Dear Mr. Monty,

I am pleased to present a proposal aimed at resolving the challenges confronted by NaN University's Special Collections library department in the digitization process of their historical book collection. Through our discussions we have identified a critical obstacle: the accurate transcription of handwritten publication dates from scanned pages, which significantly hinders the efficient sorting and digitization procedure due to the large quantity of books.

Understanding the significance of accurate and efficient preserving and facilitating access to these historic volumes, particularly by sorting them according to publication dates, we acknowledge the pressing need for an automated solution to extract these dates accurately and efficiently.

In response to this challenge, our proposed solution revolves around the implementation of a machine learning application specifically designed to interpret handwritten numbers. By employing a sophisticated classification machine learning algorithm, the application processes input images of book pages containing publication dates, accurately detecting and outputting the corresponding dates. This automation process aspires to accelerate the transcription process, ensuring precision in digitizing the books' publication dates.

The advantages of our proposed solution for NaN University are numerous. Primarily, it will significantly reduce the time and effort required for digitization, thereby streamlining operations, and increasing the productivity of the special collections department. Furthermore, the digitized dataset will facilitate effortless retrieval and organization of book records, fostering accessibility and promoting scholarly research and engagement. Additionally, our solution eliminates the guesswork involved in deciphering illegible publication dates, providing informed detection, and alleviating the challenges associated with unclear transcriptions.

Regarding costs and timeline, our phased implementation plan entails an initial investment in software development and testing, followed by deployment and training phases. While there is a moderate upfront investment, the long-term benefits, such as increased efficiency and resource allocation, will outweigh these costs. The projected completion timeline for the project is estimated at three months, during which we will conduct comprehensive model training and evaluation and finalize the application's user interface. Furthermore, we are committed to upholding the highest ethical standards throughout the project implementation by prioritizing the protection of sensitive data related to the library records.

As a team with extensive expertise in machine learning, data analytics, and software development, we are poised to deliver a tailored solution that precisely aligns with the specialized requirements of NaN University's special collections department. Using my experience as part of teams responsible for pioneering AI (Artificial Intelligence) projects, including Tesla's self-driving cars, I can confidently assure you of the quality and efficacy of our proposed solution.

Thank you for considering our proposal. I am looking forward to the transformative potential of this initiative to yield tangible benefits for our organization and the special collections department.

Sincerely,

John Huang

John Huang, Certified Machine Learning Expert

Part B: Project Proposal Plan

Project Summary

NaN University's special collections department is faced with the laborious task of digitizing handwritten dates from scanned book pages. The absence of an automated method to discern handwritten publication dates from these old books creates a notable bottleneck, impeding their ability to efficiently organize and provide easy access to these valuable resources. This project aims to deliver a machine learning application capable of interpreting handwritten numbers from images. Users will be able to upload an image and receive a numerical output. Additionally, comprehensive documentation, including a user guide, will be provided to facilitate the adoption and utilization of the application by the special collections staff. The implementation of our machine learning application offers numerous benefits to the client. Firstly, the automation of the transcription process will reduce the time and effort required for digitization, resulting in increased operational efficiency and productivity within the special collections department. Moreover, the digitized dataset will allow effortless retrieval and organization of book records by sorting the books by their publication date, which enhances accessibility for individuals interested in the books. Finally, the utilization of the machine learning application will eliminate the guesswork involved in deciphering illegible handwriting caused by poor penmanship or deteriorated paper, ensuring accurate detection.

Data Summary

The raw data for the project will be sourced from Kaggle, where a user has uploaded a transformed version of the MNIST dataset, a public dataset for handwritten digit recognition, into a CSV format [1]. This dataset contains a label as the first column and the rest of the columns are the pixel values for that number. Throughout the application development lifecycle, the data will be processed and managed, including analysis during the design phase to inform decisions regarding feature engineering, model selection, and evaluation metrics to measure success. In the development phase, the dataset will be preprocessed to be used to train the machine learning model, and during maintenance, ongoing monitoring and updates may be necessary. The MNIST dataset is well-suited for the project due to its relevance, large data size, labeled examples, and quality. Data anomalies such as outliers or missing values would be handled by removing the rows that contain the data anomalies. Since MNIST is a publicly available dataset commonly used for research purposes and contains no identifiable information, there are no notable ethical or legal concerns associated with its use.

Implementation

The project will follow the industry-standard methodology of CRISP-DM (Cross-Industry Standard Process for Data Mining). CRISP-DM consists of six phases: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment [2]. Each phase involves certain tasks aimed at sequentially guiding a project from the beginning to completion, ensuring that business objectives are addressed. CRISP-DM is beneficial for this project because the project requirements are well-defined, and the objectives are clear.

Implementation Plan:

Business Understanding:

- Conduct stakeholder interviews and requirements gathering sessions to define project objectives and success criteria.

Data Understanding:

- Acquire raw data from Kaggle and understand the characteristics of the dataset.

Data Preparation:

- Preprocess the raw data by removing any data anomalies, if applicable
- Split the dataset into training and testing subsets

Modeling:

- Train and fine tune performance of classification algorithms on the training dataset to predict handwritten numbers accurately

Evaluation:

- Evaluate model performance using classification reports, containing metrics such as accuracy, precision, recall, and F1-score
- Validate models on the testing dataset to assess generalization performance

Deployment:

- Deploy the trained machine learning model with an interactive user interface
- Allow users to upload scanned images and receive predictions of handwritten publication dates

Timeline

Milestone or deliverable	Tasks	Duration	Projected Dates
Business Understanding	Define project objectives Meet with stakeholders Define success criteria Identify Key Performance Indicators (KPIs)	1 week	06/01/24 - 06/08/24
Data Understanding	Gather relevant data sources Perform data analysis to evaluate quality of data	3 weeks	06/08/24 - 06/30/24
Data Preparation	Handle missing values, outliers, or inconsistencies in data Split the dataset into training and testing subsets	1 month	07/01/24 - 07/31/24
Modeling	Train machine learning models Refine the models based on performance metrics and feedback from stakeholders	2 weeks	08/01/24 - 08/15/24
Evaluation	Evaluate the performance of the developed models against success criteria Determine if model needs more iterations for refinement	1 week	08/16/24 - 08/23/24

Deployment	Prepare models for deployment in the production environment. Integrate model into special collections department Develop documentation and user guides	1 week	08/24/24 - 08/30/24
Monitoring and Maintenance	Track performance of the application Create protocols for ongoing maintenance, updates, and improvements based on feedback	Ongoing	N/A

Evaluation Plan

Verification Methods:

- **Data Preprocessing Verification:** Ensure that the data preprocessing is thorough and accurate, addressing any data anomalies or missing values, by removing them from the dataset. Additionally, confirm that the user's input is appropriately formatted to be detected by the machine learning model. Ensure that the data preprocessing is thorough and accurate, addressing any anomalies or missing values. Additionally, confirm that the user's image input is appropriately formatted to be detected by the machine learning model.

- **Integration Testing:** Test how different parts of the system work together, such as data processing, model training, and predictions. Make sure data pipelines send the right information to the model and handle special cases well. Confirm that the model works smoothly with user interface.

- **End-to-End Testing:** Perform thorough testing to ensure the system's functionality across all stages, including proper model training and accurate prediction output when provided with an image input of a number.

- **UI Testing**: Ensure the user interface is intuitive and user-friendly, allowing for easy comprehension. Provide a detailed user guide that enables users to navigate the process without prior knowledge. Include contact information for assistance with any questions about the system.

Validation Methods:

- **Classification Report:** Evaluation metrics to assess the effectiveness and performance of the machine learning model. This includes accuracy, precision, recall, F1-score, and frequency of each digit tested.

- **Confusion Matrix:** evaluates the model's performance. Each of the cells in the matrix are representative of the number of true positives, true negatives, false positives, and false negatives. It offers a breakdown of the model's performance across each number classification, which helps in determining strengths and weaknesses.

- **Train and Test Split:** The dataset is divided into two parts: the training set and the test set. The model is trained on the training set to learn the pixel pattern of each number. Then its performance is assessed using the test set, which serves as unseen data to evaluate the model's ability to generalize to new information.

Resources and Costs

Hardware and Software Costs:

- Hardware:
    - Laptop or desktop for model training and testing: $2000
- Software:
    - Machine learning libraries and frameworks (scikit-learn): $0
    - Development tools (Jupyter Notebook): $0
    - Git Version Control: $0
    - Cloud computing infrastructure for deployment and hosting: $100/month

Estimated Labor Time and Costs:

- Data Collection and Preparation:
    - Data collection: 1 hour, $0
    - Data preprocessing: 2 hours, $0
- Model Development and Testing:
    - Algorithm development: 1 hour, $100
    - Model training and tuning: 5 hours, $100
    - Testing and validation: 5 hours, $100
- Software Development:
    - Application development: 2 hours, $100
    - UI design and implementation: 2 hours, $100
- Deployment and Maintenance:
    - Deployment setup: 5 hours, $100

    - Hosting and maintenance: 10 hours/month, $100/month

Estimated Environment Costs:

- Maintenance:
    - Technical support and troubleshooting: $100/month
    - Software updates and patches: $100/month
- Miscellaneous:
    - Contingency budget for unforeseen expenses: $1000
    - Labor: $1000

Part D: Post-implementation Report

Solution Summary

The project addressed the labor-intensive task of digitizing handwritten dates from scanned book pages in NaN University's special collections library by providing an initial machine learning model capable of detecting and outputting handwritten digits. This manual transcription process significantly slowed down sorting and digitization efforts. To tackle this issue, a machine learning solution was proposed, which utilized the MNIST dataset for handwritten digit recognition. A classification machine learning algorithm was developed employing scikit's K-Nearest Neighbors (KNN) algorithm, where it took the input of an image and outputted the detected number.

The application effectively addresses the problem by accurately detecting and outputting the handwritten numbers depicted in the input images. While the special collections department initially requires a basic model for handwritten image detection, this functionality serves as a fundamental stepping stone towards resolving their overarching issue. By inputting scanned images of pages containing publication dates, the machine learning algorithm seamlessly identifies and outputs the detected dates, eliminating the need for manual data entry onto spreadsheets. This automation streamlines the workflow of the special collections department, reallocating resources and enhancing overall productivity.

Data Summary

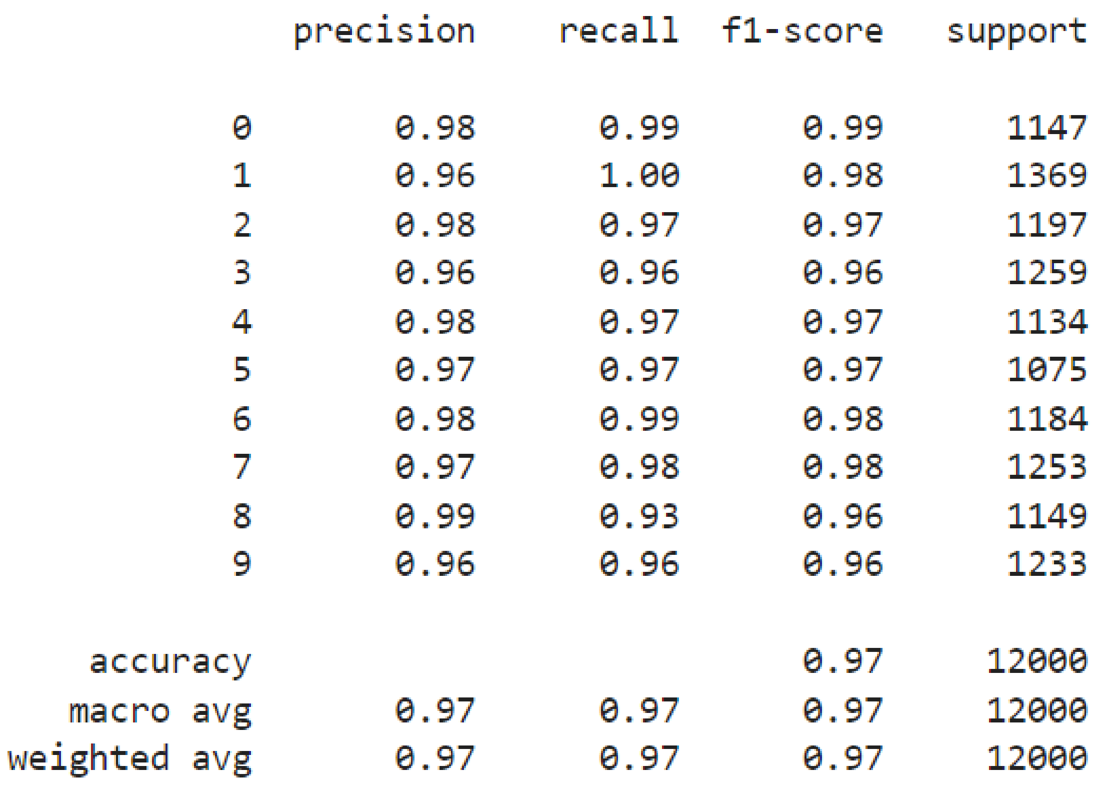
The project's raw data was sourced from Kaggle, where a user had reformatted the MNIST dataset into a CSV file. This dataset comprises labeled images of handwritten digits ranging from 0 to 9, with each image represented by pixel values. Throughout the application development process, the data underwent several stages of processing and management. During the design phase, exploratory data analysis was conducted to understand dataset characteristics and guide feature engineering decisions. In the development phase, the dataset underwent preprocessing steps such as normalization of pixel values, removal of missing values or anomalies, and division into training and testing sets. Model training utilized the training data, and model performance was assessed using the testing dataset. Maintenance activities involved continuous monitoring of model performance and potential updates to adapt to evolving data patterns. Version control systems were employed to track codebase modifications and manage development iterations effectively.

Machine Learning

Our product utilizes machine learning to automate the digitization of handwritten dates from scanned book pages in NaN University's special collections library. The K-Nearest Neighbors (KNN) algorithm, implemented with the scikit-learn library in Python, interprets handwritten numbers from images, facilitating accurate digitization. The K-Nearest Neighbors (KNN) algorithm, used for classification, predicts a data point's class based on the majority class of its nearest neighbors (with k=3 in our case). The model was trained on the training data using the fit() function, learning the relationships between input features and digit labels. Evaluation on test data assessed its performance and generalization abilities. KNN was chosen for its simplicity and suitability for image recognition tasks.

Validation

Train-test split is the validation method where the dataset is divided into training and testing sets and is the chosen validation method for the KNN model. The training set is used to train the model, while the testing set, unseen by the model, evaluates its ability to generalize to new data. The classification report provides detailed metrics such as precision, recall, accuracy, and F1 scores, offering insights into the model's performance.



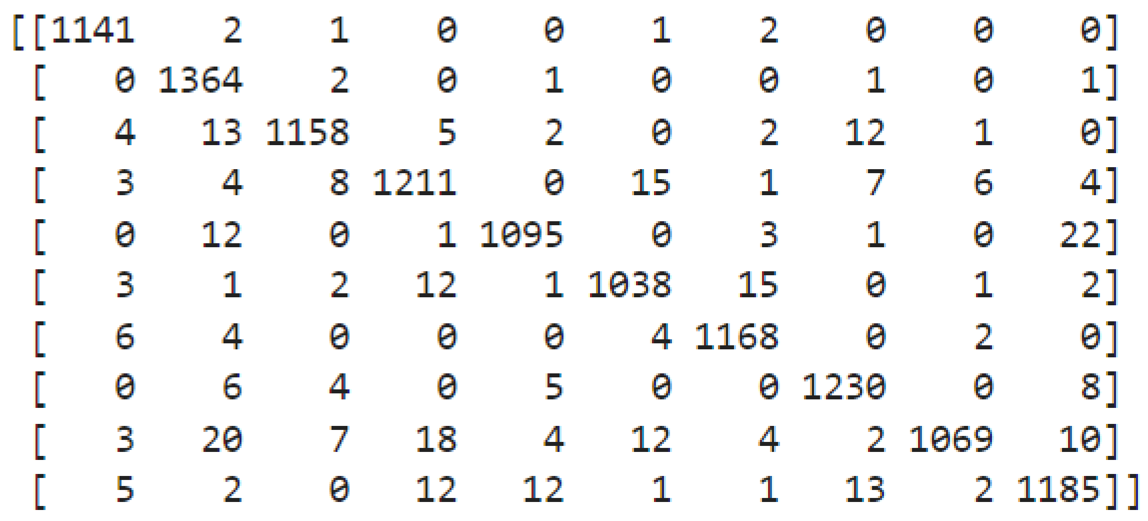
```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      1147
           1       0.96      1.00      0.98      1369
           2       0.98      0.97      0.97      1197
           3       0.96      0.96      0.96      1259
           4       0.98      0.97      0.97      1134
           5       0.97      0.97      0.97      1075
           6       0.98      0.99      0.98      1184
           7       0.97      0.98      0.98      1253
           8       0.99      0.93      0.96      1149
           9       0.96      0.96      0.96      1233

    accuracy                           0.97     12000
   macro avg       0.97      0.97      0.97     12000
weighted avg       0.97      0.97      0.97     12000
```

The classification report provides the results of the validation method. With an overall accuracy of 97% on a test dataset of 12,000 digits, the model demonstrates strong predictive capability. F1 scores exceeding 95% across all digits indicate high precision and recall. However, the recall ratio for the number 8 falls slightly to 93%, suggesting a weakness in accurately identifying this digit. This is corroborated by the confusion matrix below, which reveals frequent misclassifications of 8 as 1. Despite this, the model exhibits robust generalization to new data and maintains high accuracy overall.

Visualizations

The descriptive methods are also included in the training.ipynb notebook, but there is no explanation, they will be explained in this section here:

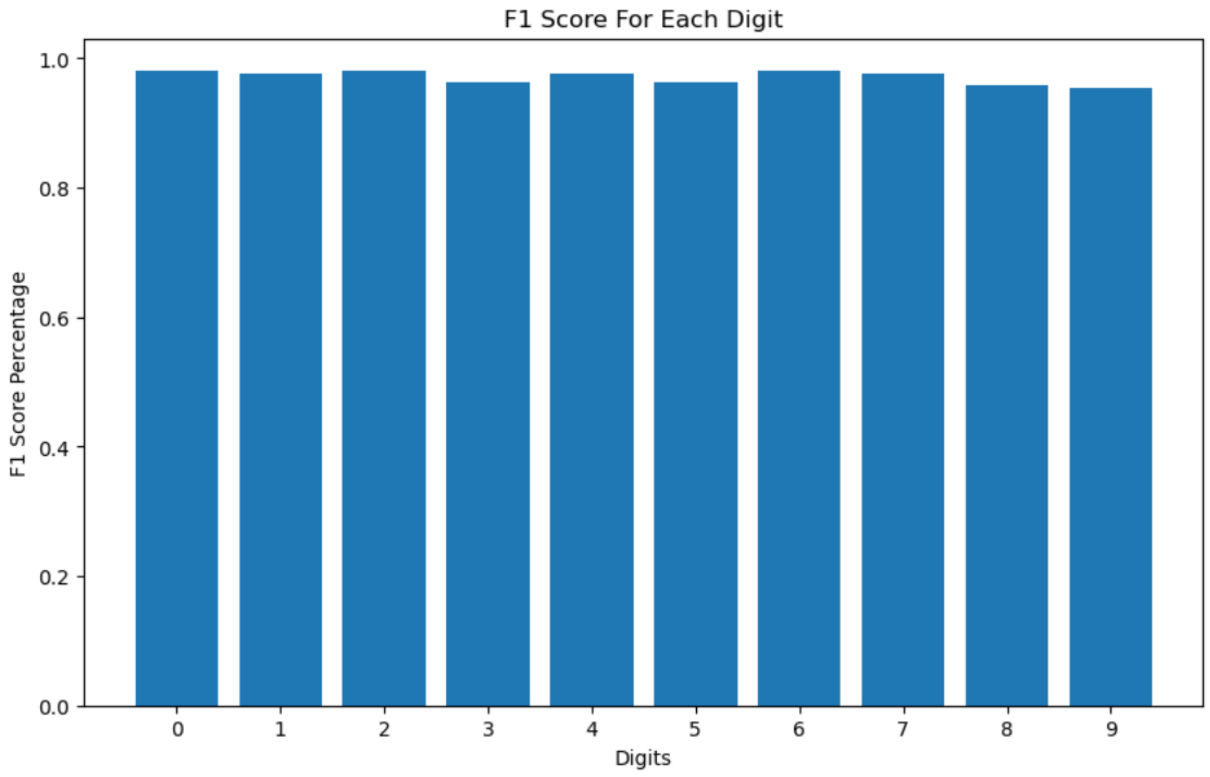
Confusion Matrix



```
[[1141    2    1    0    0    1    2    0    0    0]
 [   0 1364    2    0    1    0    0    1    0    1]
 [   4   13 1158    5    2    0    2   12    1    0]
 [   3    4    8 1211    0   15    1    7    6    4]
 [   0   12    0    1 1095    0    3    1    0   22]
 [   3    1    2   12    1 1038   15    0    1    2]
 [   6    4    0    0    0    4 1168    0    2    0]
 [   0    6    4    0    5    0    0 1230    0    8]
 [   3   20    7   18    4   12    4    2 1069   10]
 [   5    2    0   12   12    1    1   13    2 1185]]
```

This confusion matrix, organized in a 10 x 10 grid representing digits from 0 to 9, offers insights into the model's performance. Along the diagonal from top left to bottom right, each cell shows the instances where the actual digit matches the predicted digit. Moving off the diagonal, each cell represents instances where the model misclassified the digit. For instance, the value 20 in row 9 and column 1 indicates how many times the model mistakenly classified the digit 8 as the digit 1. By analyzing these off-diagonal cells, we can identify which digits are commonly confused with each other and can prompt further investigation into the features contributing to these misclassifications, giving insight on potential areas for model improvement

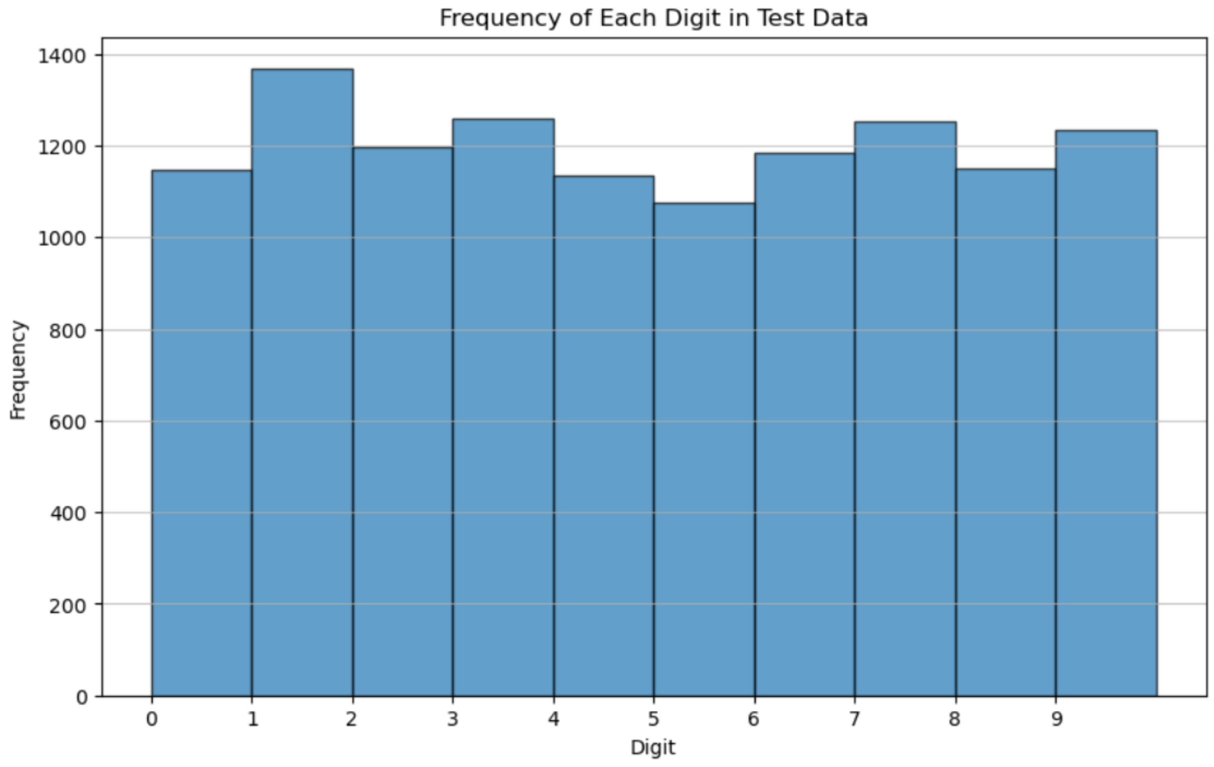
F1 Score of Each digit



This figure indicates that the F1 scores for each digit are high, all above 95%, implying robust performance across all digits. The F1 score is calculated by considering both precision (the ratio of correctly predicted positive instances to all predicted positive instances) and recall (the ratio of correctly predicted positive instances to all actual positive instances). A high F1 score suggests that the model achieves high precision and recall simultaneously.

$$\frac{\textit{Number of True Positives(TP)}}{\textit{Number of True Positives(TP)} + \textit{Number of False Positives(FP)}}$$

Frequency Distribution of Each Digit



This histogram illustrates the frequency of each digit detected during testing. Overall, the distribution seems uniform, with a discrepancy of around 200 frequencies between the most frequent digit, 1, and the least frequent, 5. Out of the total 12,000 numbers tested, this uniformity suggests that each digit received equal testing exposure. This balanced distribution is advantageous as it ensures that the model's results were not significantly influenced by the frequency of each digit.

User Guide

1. If miniconda or anaconda is already installed, skip to step 3

2. Installing miniconda
    a. Go to https://docs.anaconda.com/free/miniconda/miniconda-other-installer-links/
    b. Click on the Miniconda3 Windows 64-bit link to install Python 3.12 or the latest link
    c. Go through the installation process
    d. Keep all the options on the default setting

3. Download the Capstone C964.zip file containing the application and drag onto the desktop

4. Right click on the zip file and press "extract all" and click "extract" to move it to the desktop

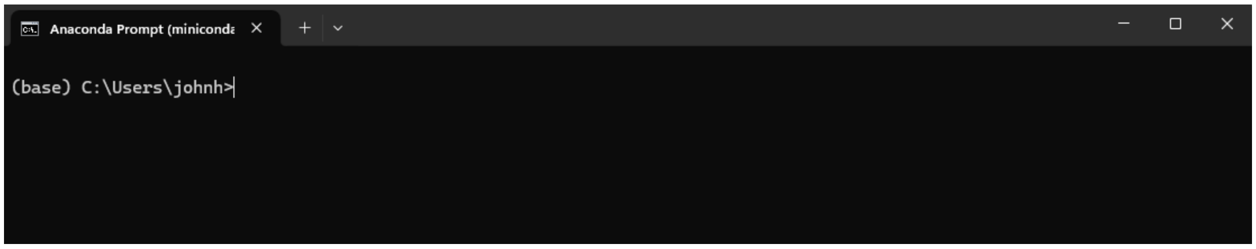
5. You can now drag the zipped file (folder with the zipper on it) to the recycle bin

6. Right click on the Capstone C964 folder and click the option that says "copy as path"

7. Go to the window search bar in your windows manager or the search bar on the bottom left

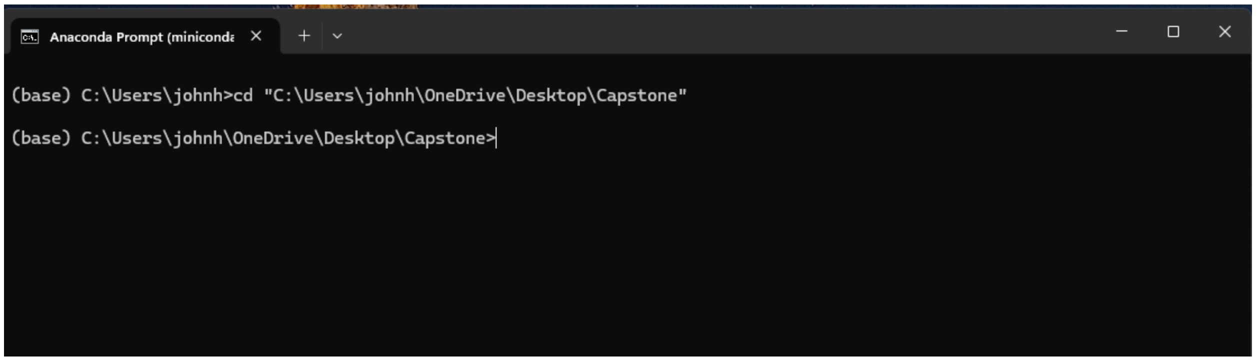
8. Type in "Anaconda" and click on the prompt that says "Anaconda Prompt (miniconda3)"

9. The command terminal that pops up should look like the image below (the johnh portion will be the user's name and will vary):



```
(base) C:\Users\johnh>
```

10. In the command terminal, type **cd** and insert a space, press **CTRL + V** to paste the path you copied earlier, and press enter



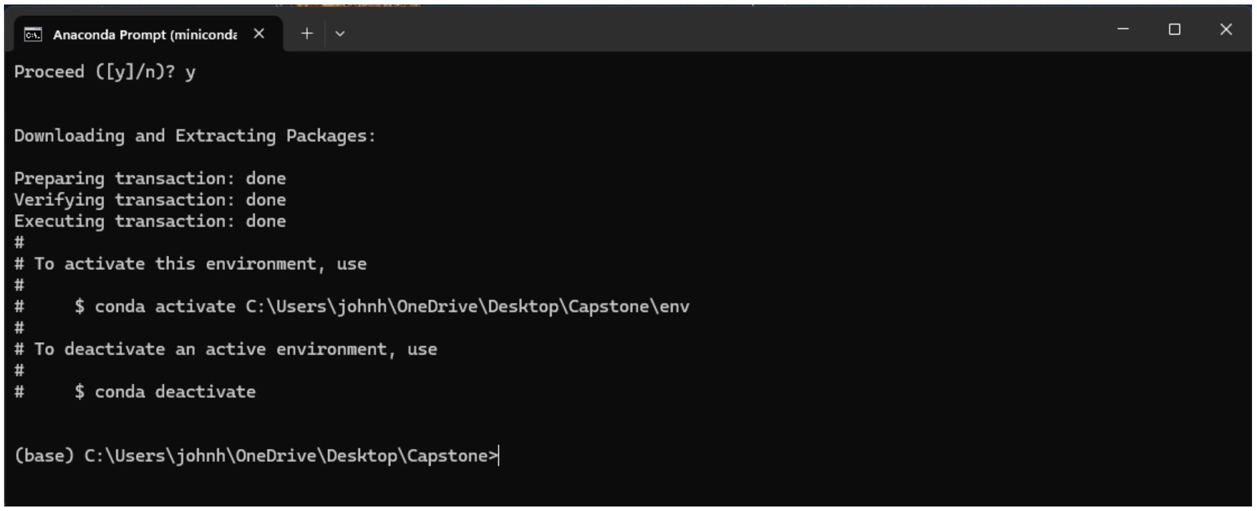
```
(base) C:\Users\johnh>cd "C:\Users\johnh\OneDrive\Desktop\Capstone"

(base) C:\Users\johnh\OneDrive\Desktop\Capstone>
```

11. Now create a new folder called env and download all the necessary dependencies/tools into that folder by using this command (copy and paste it into the terminal): **conda create --prefix ./env pandas numpy matplotlib scikit-learn jupyter** and then press enter

12. When you come across the prompt "Proceed ([y]/n)?", type "y" and enter to allow the download (it will take a minute or two depending on network speed)

13. You will come across this prompt:

    "To activate this environment, use
        $ conda activate _________"



```
Proceed ([y]/n)? y


Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate C:\Users\johnh\OneDrive\Desktop\Capstone\env
#
# To deactivate an active environment, use
#
#     $ conda deactivate


(base) C:\Users\johnh\OneDrive\Desktop\Capstone>
```

14. Copy (CTRL + C) and paste (CTRL + V) the activate line (minus the $) into the terminal and press enter

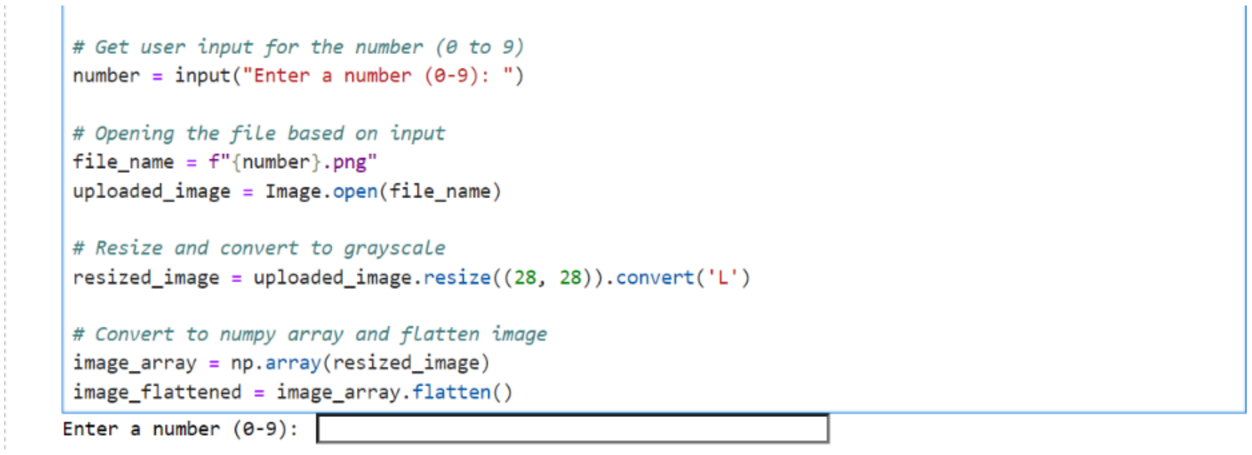
15. Then type **jupyter notebook** into the terminal and press enter to open the jupyter notebook which contains the application in the browser

16. You will see a bunch of files and images, but you only need to double click on the one that says "application.ipynb"

17. You are going to see a few boxes/cells. Click on the first box containing the import statements and press shift + enter or the play/run button on top to run the box

18. Now click on the second box with the statement “neigh = load('trained_knn_model.joblib')” and press run or shift + enter

19. Click on the box under the heading “SAMPLE IMAGE INPUT AND PREDICTION" and press shift + enter or the play/run button on top to run the box, you will receive an area to enter a number under the box as seen in this image below:

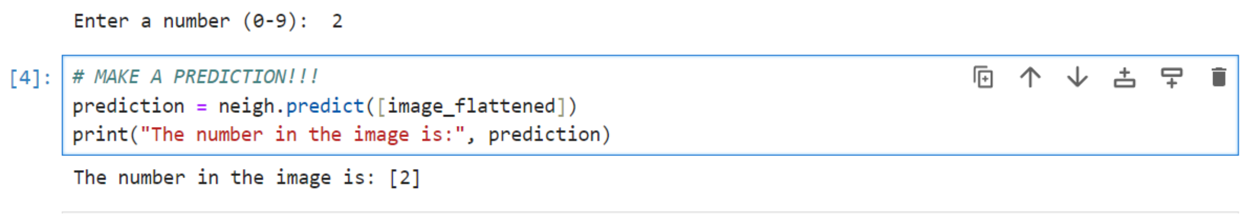


```
# Get user input for the number (0 to 9)
number = input("Enter a number (0-9): ")

# Opening the file based on input
file_name = f"{number}.png"
uploaded_image = Image.open(file_name)

# Resize and convert to grayscale
resized_image = uploaded_image.resize((28, 28)).convert('L')

# Convert to numpy array and flatten image
image_array = np.array(resized_image)
image_flattened = image_array.flatten()
Enter a number (0-9):
```

20. For a sample demonstration, I have included images of the numbers 0-9 in the application, so you can type in the image you want to use by typing a number 0-9 in the rectangle below next to “Enter a number (0-9):” and then press enter

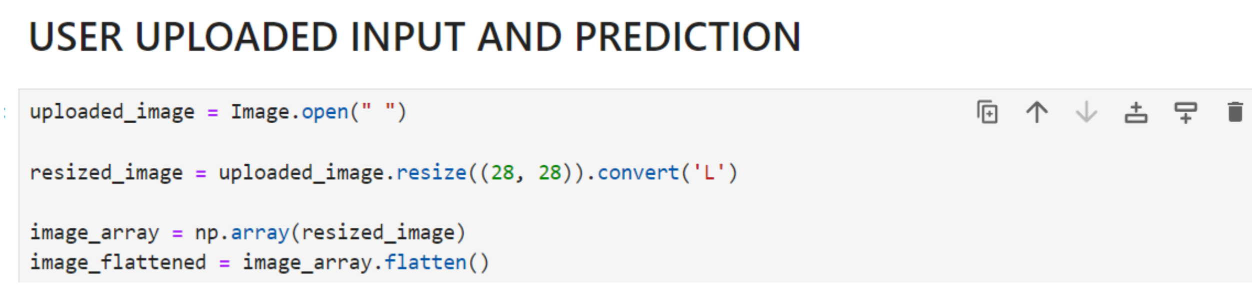
21. Next go to the box right underneath that says, “MAKE A SAMPLE PREDICTION!!!” and run that box as well using shift + enter

22. Then right underneath that box, there will be a sentence that says “The number in the image is: [predicted number]”:



```
Enter a number (0-9):  2

[4]: # MAKE A PREDICTION!!!
     prediction = neigh.predict([image_flattened])
     print("The number in the image is:", prediction)

The number in the image is: [2]
```

23. If you want to use your own images, you can download the image and drag it into the Capstone C964 folder that is on your desktop, then you would go to the box underneath the heading: “USER UPLOADED INPUT AND PREDICTION”



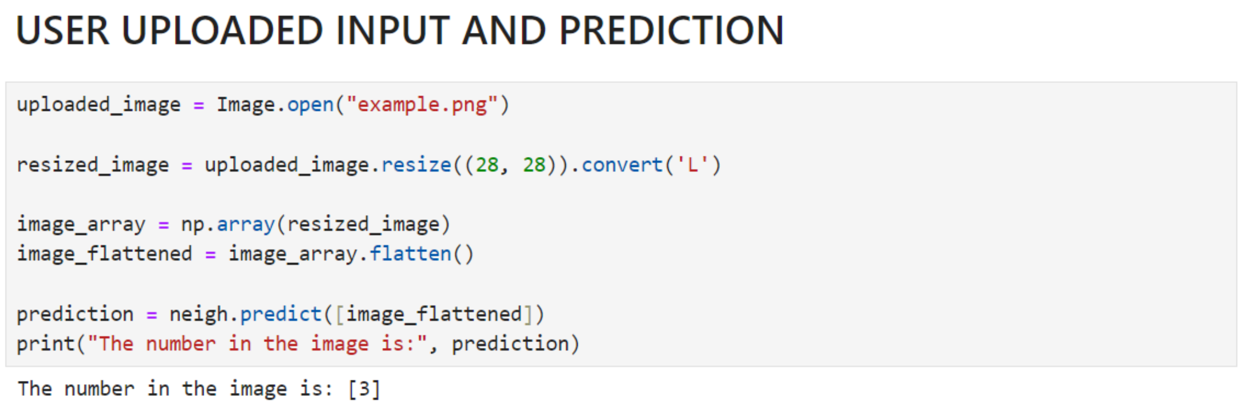
USER UPLOADED INPUT AND PREDICTION

```
uploaded_image = Image.open(" ")

resized_image = uploaded_image.resize((28, 28)).convert('L')

image_array = np.array(resized_image)
image_flattened = image_array.flatten()
```

24. On the line that says: uploaded_image = Image.open(" ")

    a. You would type in the name of the image that you want to predict, including the type of image (i.e., png, jpeg, etc.) and then you run the code in the box

    b. So, for example if your image is called “example.png”, then the line would look similar to the screenshot below:



USER UPLOADED INPUT AND PREDICTION

```
uploaded_image = Image.open("example.png")

resized_image = uploaded_image.resize((28, 28)).convert('L')

image_array = np.array(resized_image)
image_flattened = image_array.flatten()

prediction = neigh.predict([image_flattened])
print("The number in the image is:", prediction)
```

```
The number in the image is: [3]
```

25. To exit from the instance, pull up your command terminal and press **control + c**

26. Then type in **conda deactivate** and press enter to close miniconda

27. To boot up the jupyter notebook again, follow steps 13, 14, and 15

Reference Page

1. *MNIST in CSV*. (n.d.). Www.kaggle.com. Retrieved May 10, 2024, from http://www.kaggle.com/datasets/oddrationale/mnist-in-csv

2. Hotz, N. (2023, January 19). *What is CRISP DM?* Data Science Project Management. https://www.datascience-pm.com/crisp-dm-2/

3. scikit-learn. (2019). *sklearn.neighbors.KNeighborsClassifier — scikit-learn 0.22.1 documentation*. Scikit-Learn.org. Retrieved May 11, 2024, from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html