Machine Learning

Kunskapskontroll 2

Anna Strbac

EC Utbildning

202403

Abstract

The aim of the study is to develop an application capable of receiving loaded images containing handwritten digits as input. These images will be processed, and employing a machine learning model, the application will predict the handwritten digit depicted in the image.

Leveraging the well-known MNIST dataset, which consists of extensive collection of handwritten digits, this study will explore and assess various machine learning algorithms for their effectiveness in digit classification tasks.

The study explores a range of machine learning algorithms, including SVC (Support Vector Classifier), KNN (K-Neighbors Classifier) and Extra Trees Classifier, to identify the one with the highest performance. The accuracy metric will be used to identify the most effective algorithm. The top-performing algorithm will be implemented to predict handwritten digits.

Some techniques for image preprocessing will be used to achieve better accuracy in predicting custom handwritten digits.

Keywords: Digit Recognition, Handwritten digits, MNIST Dataset, Machine Learning Algorithms, Digit Classification Task.

1 Introduction

This study will primarily explore handwritten digits recognition, a significant domain within machine learning research and development, offering abundant opportunities for real-life applications. Handwriting digit recognition remains a highly complex task due to various factors, including handwriting variability, noise and distortion in images, and limited training data.

One of the first practical applications of handwritten digit recognition methods was the development of a system for reading ZIP codes. Digit recognition methods are also applied to solve practical tasks such as bank check processing, automated form processing, number plate recognition, postal mail sorting, etc.

In practical applications, ensuring high performance is paramount when employing digit recognition methods. Accuracy holds immense significance, as it directly influences decision-making quality. For example, in a postal code recognition system, even small errors in digit recognition can lead to misrouting of mail. This could result in delays in mail delivery or even delivery to the wrong address, leading to customer dissatisfaction and increased costs for error correction. Therefore, high accuracy of the machine learning model is crucial to ensure effective and reliable operation of the system in real life.

1.1 Objectives

This report aims to explore the MNIST dataset, experiment with various machine learning algorithms to find the most effective one, and implement it for predicting handwritten digits. Additionally, it focuses on determining the image preprocessing steps crucial for achieving high accuracy in predicting personal handwritten digits.

1.2 Research questions

To achieve this objective, the following inquiries will be addressed:

- Is it feasible to achieve an accuracy exceeding 90% in handwritten digit prediction?
- What challenges may arise when attempting to predict personal handwritten digits?
- How efficiently can the digit prediction application perform?

2 Theory

This section introduces theoretical concepts essential for contextual comprehension of this study.

2.1 Train–validation-test split methodology

Training data is used to train and make the model learn the hidden patterns in the data. It's the portion of the dataset that the model has access to during the training process. Validation data is used to adjust the model's hyperparameters and to evaluate its performance during training. Test data represents new, unseen data that the model hasn't been exposed to during training or validation. Test data is used to evaluate the model's performance after it has been trained and fine-tuned.

2.2 Data preprocessing

2.2.1 MinMaxScaler

The MinMax Scaler is a data scaling technique for normalization that sets the minimum value of a feature to zero and the maximum value to one. By compressing the data within a predefined range, typically between 0 and 1, it transforms the features to fit within this range without altering the shape of the original distribution.

2.2.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a method used to transform high-dimensional data into a lower-dimensional while preserving as much information as possible. It is widely employed in exploratory data analysis, visualization, and data preprocessing. PCA is extremely useful when working with data sets that have a lot of features.

2.2.3 Pipeline

A pipeline in machine learning is a series of data processing steps chained together sequentially. It strings together different tasks, like preparing the data and building models, in a smooth sequence. This structured approach makes it easier to develop and deploy models efficiently.

2.3 Performance Measures

2.3.1 Accuracy

Accuracy is a performance metric for evaluating classification models, representing the percentage of all observations correctly classified by the model.

2.4 Machine Learning Models

2.4.1 Support Vector Machine (SVM)

A Support Vector Machine (SVM) is a very powerful and versatile supervised machine learning algorithm, capable of performing linear or nonlinear classification, regression, and even outlier detection (Géron, 2019, p. 155).

The main objective of the SVM algorithm is to find the best hyperplane to divide the dataset. The hyperplane aims to maximize the margin between the nearest points of different classes. The dimension of the hyperplane depends upon the number of features. In two-dimensional space, a hyperplane is a line, while in a three-dimensional space, it is a plane. The closest points to the hyperplane are called support vectors. Margin refers to the space between the hyperplane and the closest data point on either side.

Support Vector Machine (SVM) utilizes hyperparameters like the regularization parameter C, which is used for controlling the trade-off between maximizing the margin and minimizing the classification error, as well as other hyperparameters such as the choice of kernel used to transform data into a higher-dimensional space and the gamma parameter , which affects the complexity of decision boundaries for non-linear kernels.

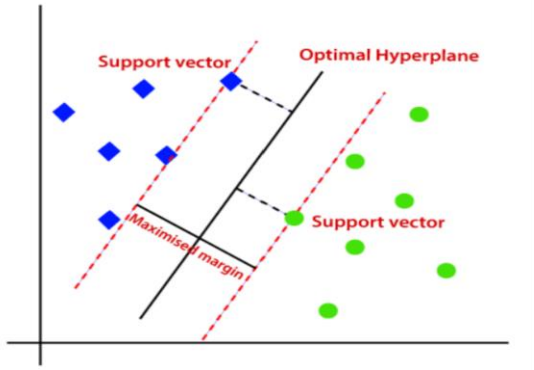

Figure 1: SVM visualization

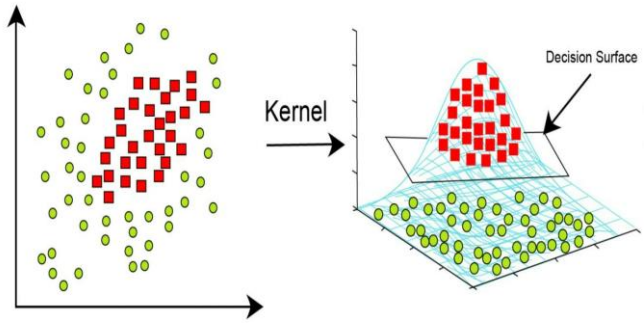

Figure 2: Kernel visualization

2.4.2 K-Nearest Neighbors (K-NN)

The k-nearest neighbors (K-NN) algorithm is a supervised machine learning versatile algorithm that can be used for both classification and regression. The algorithm is widely used because it is simple and easy to implement, and it does not require any assumptions about the underlying data distribution. K-NN is less sensitive to outliers compared to other algorithms.

The K-NN algorithm functions by identifying the K closest data points to a target point, typically calculated using a distance measure like Euclidean distance. Once these nearest neighbors are found, the algorithm decides the class or value of the target point based on the most common class (for classification tasks) or the average value (for regression tasks) among its neighbors. This approach allows

the algorithm to adapt to different patterns and make predictions based on the local structure of the data.

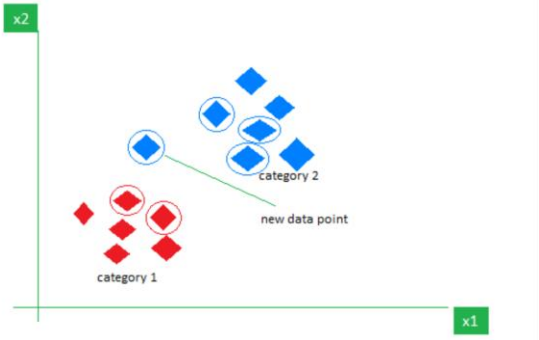

Figure 3: KNN visualization

2.4.3 Extra Trees Classifier

The Extra Trees classifier, also known as Extremely Randomized Trees, is an ensemble learning method based on decision trees. It builds multiple decision trees from randomly selected subsets of the training data and randomly selects feature subsets at each node. This randomness reduces overfitting and increases diversity among the trees. The final prediction is made by averaging the predictions of all trees in the ensemble.

2.4.4 Grid Search CV

Grid Search CV utilizes cross-validation to assess how well each hyperparameter combination performs. It divides the training data into several subsets, then trains the model on various hyperparameter combinations while evaluating its performance on the validation set. It aims to find the combination that maximizes classifiers performance on the training data, ultimately leading to improved generalization and predictive accuracy on unseen data.

3 Metod

In this chapter, we delve into data exploration, model selection, and the techniques used for image preprocessing.

3.1 Tools

The programming was conducted using scikit-learn, a Python-based machine learning library. OpenCV (open-source computer vision) was used for personal image processing. A digit prediction application was developed using Streamlit (an open-source Python library used for building web applications for machine learning).

3.2 Data Collection

The dataset MNIST is loaded using the fetch_openml function from the sklearn.datasets module. After loading, the data is separated into features and target labels. The features are stored in variable X, while the target labels are stored in variable y.

3.3 Data Exploration

MNIST dataset is a large set of handwritten digits that is commonly used for training various image processing systems [1]. It is a subset of a larger NIST Special Database 3 (digits written by employees of the United States Census Bureau) and Special Database 1 (digits written by high school students). MNIST contains 70,000 small monochrome images of handwritten digits written by 250 different individuals. Each image is labeled with the digit it represents.

The original black and white (bilevel) images from NIST were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. The images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field [2].

Each image in the MNIST dataset contains 784 features, representing a resolution of 28x28 pixels. Each feature simply represents one pixel's intensity, from 0 (white) to 255 (black) (Géron, 2019, p. 87-88).

The MNIST dataset comes pre-shuffled. It contains 60,000 training images and 10,000 testing images.

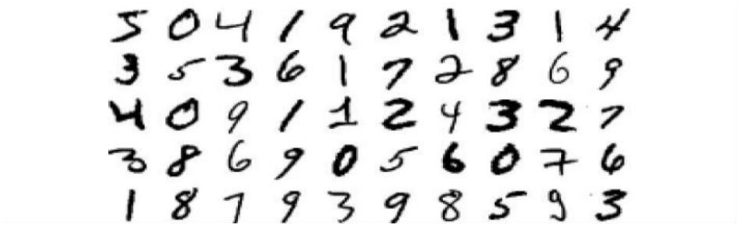

Figure 4: Examples for MNIST data

A few images from the MNIST dataset are showcased in the picture below to provide an understanding of the complexity of the classification task.

Exploratory Data Analysis (EDA) was conducted to understand the dataset being handled, particularly to visualize how input data should appear for personal handwritten digit prediction. Additionally, it aimed to analyze the writing styles of individuals who contributed to the dataset, helping to understand the optimal format for input images to enhance prediction accuracy.

3.4 Training and Evaluation

In this study, we utilize the train-validation-test split methodology, dividing the dataset into three essential subsets: the training set, the validation set, and the test set.

The split uses a random process to divide the data. In this methodology, the random state parameter ensures reproducibility by fixing the random seed used for the splitting process.

3.5 Data preprocessing

The MinMaxScaler and PCA were applied to preprocess the data when using SVM through a pipeline.

3.6 Model Implementation

Three machine learning models were evaluated: Support Vector Machines (SVM), K-Nearest Neighbors (KNN) and Extra Trees Classifier. The optimal hyperparameters for the models were selected using Grid Search CV. The selection of the best model is based on the accuracy metric.

3.7 Preparing Personal Images for Prediction

For accurate predictions of handwritten digits, it's essential to adjust the images to match the format of the dataset.

Several image processing steps have been conducted to conform to the dataset format:

1) Conversion to grayscale
2) Enhancement of brightness and contrast: Increasing brightness makes the image brighter and clearer, particularly useful for images with insufficient lighting, while increasing contrast sharpens and emphasizes the differences between dark and light areas.
3) Inversion, converting bright pixels to dark and vice versa.
4) Resizing to a 28x28 pixel dimension, aligning it with the MNIST format.
5) Application of thresholding to convert the image into a binary format: Applying thresholding can help remove background noise by simplifying the image and focusing on the main features, which can aid in better recognition by machine learning algorithms.
6) Implementation of cropping to remove any excess background. It iteratively removes rows and columns of zeros from the edges of the image until encountering non-zero pixel values.
7) Application of Gaussian blur: After applying thresholding and cropping the image, distortion might occur. Gaussian blur helps in reducing jagged edges and smoothing out transitions between different regions of the image, resulting in clearer and more visually appealing digit representations.
8) Creation of a new 28x28 image and placement of the 20x20 image in its center.
9) Shifting of a 28x28 pixel image so that its center aligns with the center of a 28x28 pixel field. This is achieved by calculating the center of mass of pixels with the center_of_mass function and then shifting the image accordingly to place the center where it needs to be.
10) Reshaping into a one-dimensional array with a single row.

4 Results and Discussion

In this chapter, we present a comparison of the models' performance and highlight the challenges that may arise in handwritten digit recognition.

4.1 Model selection

Three different machine learning models were tested on validation set, all demonstrating impressively high accuracy scores, each exceeding 96%.

Accuracy for different models	
Support Vector Classifier	98,68 %
K-Nearest Neighbors (KNN)	97,44 %
Extra Trees Classifier	96,77 %

Tabell 1: Accuracy for the examined models

The SVM model demonstrated the highest accuracy, achieving 98.68%. Given its superior performance, the model was trained on the train-validation set and subsequently evaluated on the test set. The accuracy on the test set reached 98.34%. Following that, the model was applied to predict custom handwritten digits.

4.2 Difficulties in predicting custom handwritten digits

In order to utilize personal handwritten digits for prediction, it was crucial to adapt them to the dataset format. Without image preprocessing, achieving high accuracy in predicting custom handwritten digits would be impossible.

Through testing various image loads, it becomes evident that prediction accuracy is significantly influenced by the quality of the loaded image, the preprocessing steps applied to it and the style in which the digit is written.

Below, we can observe an example demonstrating how image preprocessing can impact prediction accuracy. The digit on the left was already preprocessed with all the steps described in section 3.7 except for applying the center of mass adjustment.

Figure 5: Before applying center of mass

Figure 6: After applying center of mass

We observe that without applying preprocessing step such as center of mass, the prediction was incorrect and yielded a result “2”. This discrepancy may be understandable when considering the appearance of digit 7 in the dataset.

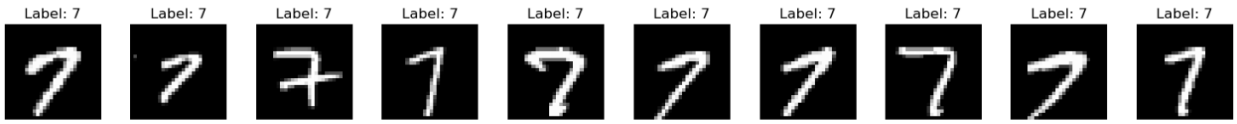

Figure 7: Representation of digit 7 in dataset

One more example which illustrates how image preprocessing choices can influence predictions. Some images may have slightly darker backgrounds, which can affect prediction accuracy. In such instances, thresholding techniques can be applied to address this issue.

In this example, thresholding values of 100 and 255 were employed to binarize the image. Notably, pixels in the bottom right corner on figure 9 exhibited values higher than 100 after inversion and thresholding. This issue led to inaccuracies in image cropping, ultimately resulting in incorrect predictions.

In contrast, when thresholding values of 127 and 255 were applied to the same image, the issue was resolved, resulting in accurate predictions.

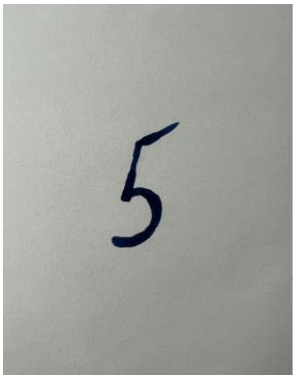

Figure 8: Original Image

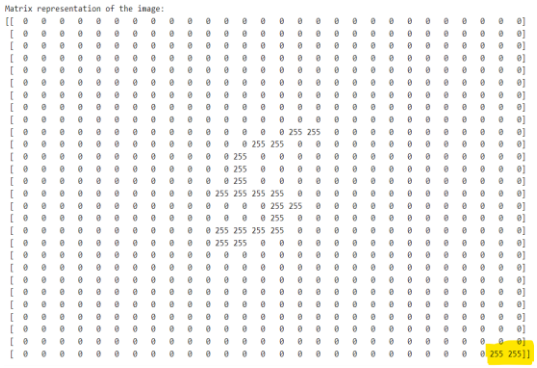

Figure 9: Matrix of resized, inverted, thresholded image

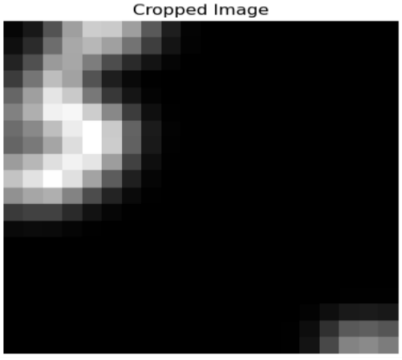

Figure 10: Cropped image

The other factor that can influence the prediction is the quality of the image.

In the example below, within the pixel value matrix (figure 13), we cannot clearly identify which digit it is. In this case applying thresholding will not yield results since the pixels lack sufficient strength and contrast to distinguish between the image and the background.

Figure 11: Original Image

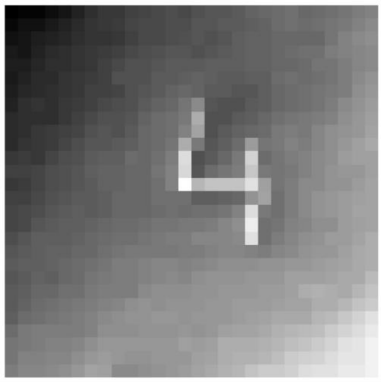

Figure 12: Image resized and inverted

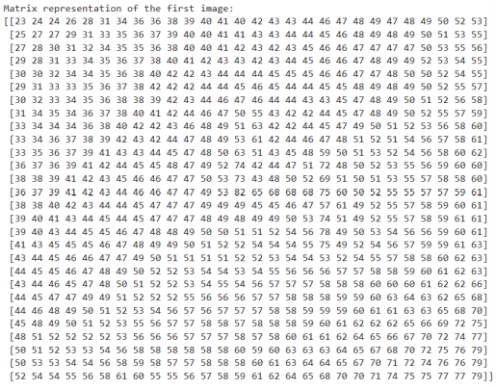

Figure 13: Matrix of resized and inverted image

Another crucial factor affecting prediction accuracy is the manner in which the digits are written. In the following example, we notice that the digit was written slightly differently from the data in the dataset (figure 16) which the models were trained on. The digit below was predicted as 3. The variations in writing style could potentially lead to incorrect predictions.

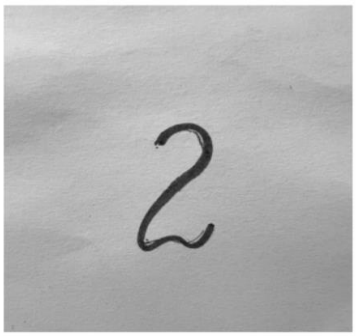

Figure 14: Original Image

Figure 15: Preprocessed Image

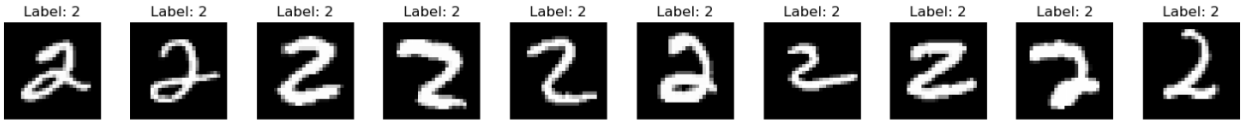

Figure 16: Representation of digit 2 in dataset

4.3 Conclusion

It can be concluded that the model we trained performs excellently on the test set of MNIST dataset, and it also works exceptionally well when the pictures are adjusted to the MNIST format precisely.

Even though applied image processing techniques can address many issues, such as when the digit isn't placed in the center of the image or when the digit doesn't conform to the dataset's digit size, predicting custom handwritten digits in practice can be challenging. When capturing images with a mobile phone or laptop camera, we often encounter background noise. This noise can pose challenges for accurate predictions, even when the model yields reasonably precise results.

Moreover, factors such as handwriting style play a significant role. The dataset used to train the model includes handwritten digits from 250 individuals, resulting in 250 different handwriting styles. In practice, people can have vastly different writing styles, potentially leading to suboptimal outcomes for our desired prediction accuracy.

For individuals unfamiliar with how the dataset appears, utilizing the Streamlit application to predict digits may frequently yield incorrect results. Therefore, specifying the expected appearance of input images in the application and providing examples of how the images should look would be beneficial for utilizing this application effectively.

References

Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. (R. R. Tache, Ed.) Canada: O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Retrieved from http://oreilly.com

[1] "MNIST_database":

https : //en.wikipedia.org/wiki/MNIST_database

[2] "MNIST database":

https : //paperswithcode.com/dataset/mnist

Teoretiska frågor

1. Kalle delar upp sin data i ”Träning”, ”Validering” och ”Test”, vad används respektive del för?

Training data is used to train and make the model learn the hidden patterns in the data. It's the portion of the dataset that the model has access to during the training process. Validation data is used to adjust the model's hyperparameters and to evaluate its performance during training. Test data represents new, unseen data that the model hasn't been exposed to during training or validation. Test data is used to evaluate the model's performance after it has been trained and fine-tuned.

2. Julia delar upp sin data i träning och test. På träningsdatan så tränar hon tre modeller; ”Linjär Regression”, ”Lasso regression” och en ”Random Forest modell”. Hur skall hon välja vilken av de tre modellerna hon skall fortsätta använda när hon inte skapat ett explicit ”valideringsdataset”?

Cross-validation method can be used to compare the performance of different models in this case. In this method, the available data is divided into multiple folds, using one of these folds as a validation set, while the others are used to train the model.

3. Vad är ”regressionsproblem? Kan du ge några exempel på modeller som används och potentiella tillämpningsområden?

Regression is a type of supervised learning, where the algorithm learns from examples where the correct answers are provided. It aims to find a relationship between independent variables and dependent variables. Regression models are used to predict numerical outcomes. For example, in economics, they can be used to predict house prices based on where the house is located, how big it is, etc. In medicine, regression models can be used to predict patients survival time or risk of complications, using information about their health history and treatment.

Some models used for regression include Linear Regression, Random Forest Regression, Support Vector Regression, and Decision Trees.

4. Hur kan du tolka RMSE och vad används det till: $RMSE$ = √∑(yi − $\hat{y}i$) 2 i

RMSE (Root Mean Square Error) measures the average difference between a model’s predicted values and the actual values. RMSE is the most common metric used to evaluate the performance of regression models. After training the regression model on the training data, we evaluate it on the validation data using RMSE. A lower RMSE means that our predictions are more accurate.

5. Vad är ”klassificieringsproblem? Kan du ge några exempel på modeller som används och potentiella tillämpningsområden? Vad är en ”Confusion Matrix”?

Classification problem is a type of supervised learning where the goal is to predict which category or class a given observation belongs to, based on its features or characteristics. Some examples of classification problems include spam filtering, where we determine if an email is spam or not, or image classification, where we assign images to predefined groups, such as bird species or different types of architectural styles.

Some models used for regression include Logistic regression, Support vector Classifier, K-nearest neighbors, Decision trees.

A confusion matrix is a performance measurement tool for machine learning classification problems. It provides a summary of prediction results by tabulating the number of correct and incorrect predictions, broken down by each class.

6. Vad är K-means modellen för något? Ge ett exempel på vad det kan tillämpas på.

K-means model is an unsupervised machine learning algorithm used to analyze clusters and partition data. The K means clustering algorithm divides a set of n observations into k clusters based on their similarities.

For example, if we have customer data, with the help of K-means, it is possible to identify groups of similar customers and then target each group with different types of marketing.

7. Förklara (gärna med ett exempel): Ordinal encoding, one-hot encoding, dummy variable encoding. Se mappen ”l8” på GitHub om du behöver repetition.

Machine learning models require all input and output variables to be numeric. So, if the dataset includes categorical data, it needs to be encoded into numerical format before training and assessing a model.

The most popular techniques for this purpose are ordinal encoding, one-hot encoding, dummy variable encoding

Ordinal encoding is used when categorical variables have an inherent order or ranking. In ordinal encoding, each unique category value is assigned an integer value. Example of ordinal encoding could be ranking age groups of individuals, where "18 to 25" = 1, "26 to 40" = 2, "41 to 55" = 3, and "56 to 70" = 4. Ordinal encoding is appropriate in this context because the age groups are inherently ordered or ranked in a meaningful way. Each age group represents a distinct category, and there is a clear progression from one category to the next based on increasing age ranges.

For non-ordinal categorical variables, using ordinal encoding might force an incorrect order, leading to poor performance. In such cases, one-hot encoding is preferred. It replaces the integer encoding with

binary variables, one for each unique value, avoiding incorrect assumptions about order. An example of one-hot encoding could be encoding the days of the week in a dataset. If a data entry corresponds to Monday, the "Monday" variable would be set to 1 and the other variables would be set to 0. Similarly to all other days of the week.

When handling categorical variables with more than two categories, dummy variables are often preferred over one-hot encoding, because they reduce redundancy in the dataset. If we recognize that [1, 0, 0] represents "blue" and [0, 1, 0] represents "green", there's no need for an extra binary variable to represent the color "red." Alternatively, we can utilize 0 values for both "blue" and "green" independently, as exemplified by [0, 0].

8. Göran påstår att datan antingen är ”ordinal” eller ”nominal”. Julia säger att detta måste tolkas. Hon ger ett exempel med att färger såsom {röd, grön, blå} generellt sett inte har någon inbördes ordning (nominal) men om du har en röd skjorta så är du vackrast på festen (ordinal) – vem har rätt?

Julia is correct. While colors are typically represented as nominal variables, they may take on an ordinal aspect in specific contexts, as exemplified in the question.

9. Kolla följande video om Streamlit: Vad är Streamlit för något och vad kan det användas till?

Streamlit is an open-source framework designed to transform data scripts into fully functional web applications. Streamlit enables building web apps using Python. It can work with different Python tools like pandas, scikit-learn, etc. Streamlit apps can perform various tasks such as data visualization, machine learning model testing, and sharing insights,

- Utmaningar du haft under arbetet samt hur du hanterat dem.

Under arbetets gång stötte jag på mycket ny information som vi inte hade täckt under lektionerna. Det var utmanande att hitta relevanta informationskällor, men till slut lyckades jag ta fram den nödvändiga informationen för att lösa uppgiften. Det var en mycket lärorik process.

- Vilket betyg du anser att du skall ha och varför.

Hade varit skönt att få VG. Trots att uppgiften var relativt komplex, lyckades jag skapa en app som predikterar handskrivna siffror med hög noggrannhet.