Number Recognition

Training a model to accurately predict handwritten numbers



Erik Fredrixon

EC Utbildning

ML Kunskapskontroll

2024–03

Abstract

The objective of this project is to create a model that can predict my handwritten numbers correctly, and then use that model to correctly predict my handwritten numbers live in a web app. I will compare different Machine Learning models and use the model with the best accuracy score in a web app using Streamlit. This project is done to create a good base understanding of how to implement all the steps from this course to complete an ML project from start to finish. And to be able to use this base understanding to complete even more complex projects in the future.

Table of Contents

[1 Introduction 1](#_Toc161941250)

[1.1 Research Questions 1](#_Toc161941251)

[2 Theory 2](#_Toc161941252)

[2.1 Evaluation Metrics 2](#_Toc161941253)

[2.1.1 Confusion Matrix 2](#_Toc161941254)

[2.1.2 Accuarcy 2](#_Toc161941255)

[2.1.3 Precision 2](#_Toc161941256)

[2.1.4 Recall 2](#_Toc161941257)

[2.1.5 F1-SCORE 2](#_Toc161941258)

[2.2 Multi-Class Classification Models 3](#_Toc161941259)

[2.2.1 Linear SVM Classification 3](#_Toc161941260)

[2.2.2 Random Forest 3](#_Toc161941261)

[2.2.3 Extra-Trees Classifier 3](#_Toc161941262)

[2.2.4 KNeigbors Classifier 3](#_Toc161941263)

[2.2.5 Logistic Regression 3](#_Toc161941264)

[2.2.6 Voting Classifier 3](#_Toc161941265)

[3 Method 4](#_Toc161941266)

[3.1 Tools 4](#_Toc161941267)

[3.2 Data Collection 4](#_Toc161941268)

[3.3 Exploratory Data Analysis (EDA) 4](#_Toc161941269)

[3.4 Image Transformation 5](#_Toc161941270)

[3.5 Model Selection 6](#_Toc161941271)

[3.6 Model Training 6](#_Toc161941272)

[3.6.1 GridSearchCV 6](#_Toc161941273)

[3.7 Handwritten Images 9](#_Toc161941274)

[3.8 Streamlit Web App 9](#_Toc161941275)

[4 Results and Discussion 10](#_Toc161941276)

[4.1 Results 10](#_Toc161941277)

[4.1.1 Model Evaluation 10](#_Toc161941278)

[4.1.2 Performance Evaluation 11](#_Toc161941279)

[4.1.3 Handwritten Images 11](#_Toc161941280)

[4.1.4 Streamlit Web App 11](#_Toc161941281)

[4.2 Discussion 12](#_Toc161941282)

[4.2.1 Model 12](#_Toc161941283)

[4.2.2 Handwritten Images 13](#_Toc161941284)

[4.2.3 Streamlit Web App 13](#_Toc161941285)

[5 Conclusions 14](#_Toc161941286)

[6 Teoretiska frågor 15](#_Toc161941287)

[7 Självutvärdering 17](#_Toc161941288)

[References 18](#_Toc161941289)

# Introduction

In everyday life, a person unknowingly uses machine learning a lot. From automated chatbots and email spam filters to the ML algorithms that govern the facial recognition we use to open our smartphones.

”[Machine learning (ML)](https://www.ibm.com/topics/machine-learning)—the [artificial intelligence (AI)](https://www.ibm.com/topics/artificial-intelligence) subfield in which machines learn from datasets and past experiences by recognizing patterns and generating predictions—is a $21 billion global industry projected to become a [$209 billion industry](https://www.globenewswire.com/news-release/2022/04/04/2415724/0/en/Machine-Learning-Market-Size-2022-2029-Worth-USD-209-91-Billion-Exhibiting-a-CAGR-of-38-8.html) by 2029.” ( [IBM Data and AI Team](https://www.ibm.com/blog/author/ibm-data-and-ai/), 2024).

The ML industry is a huge one and this course gives us the fundamentals to build ML projects like this one. The problem this project aims to address is predicting handwritten numbers in a web application.

In this project, I will use JupyterNotebook and the Python programming language to build and train different classification models. After I have trained the classification models, I will use the best and build a Python script in Microsoft Visual Studio and use the Streamlit library to create my web application.

The dataset I will be working with is the well-known MNIST dataset.

“There are 70,000 images, and each image has 784 features. This is because each image is 28 × 28 pixels, and each feature simply represents one pixel’s intensity, from 0 (white) to 255 (black).” (Géron,2019, p.86).

## Research Questions

To achieve this project’s goal, the following questions will be answered:

1. Can I create a classification model that reaches a 95% or above accuracy score on the full dataset?
2. Can I then use that model to predict my own handwritten numbers?
3. Can I use that same model to predict my own handwritten numbers live through my webcam in a web application using Streamlit?

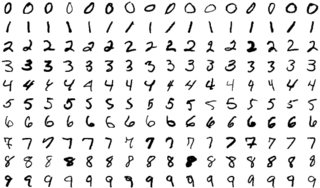


Figure 1 Digits from the MNIST dataset

# Theory

## Evaluation Metrics

### Confusion Matrix

“A **confusion matrix** is a matrix that summarizes the performance of a machine learning model on a set of test data. It is a means of displaying the number of accurate and inaccurate instances based on the model’s predictions. It is often used to measure the performance of classification models, which aim to predict a categorical label for each input instance.

The matrix displays the number of instances produced by the model on the test data.

**True positives (TP):** occur when the model accurately predicts a positive data point.

**True negatives (TN)**: occur when the model accurately predicts a negative data point.

**False positives (FP)**: occur when the model predicts a positive data point incorrectly.

**False** **negatives (FN)**: occur when the model mispredicts a negative data point.” (*Confusion Matrix in Machine Learning* 2024).

Predicted Positive Predicted Negative

Actual Positive TP FN

Actual Negative FP TN

### Accuarcy

“Put simply, classification accuracy tells us the percentage of the total classification predictions that were correct. One of the biggest drawbacks of accuracy is that it is a bad metric when you have an imbalanced dataset. When the dataset is imbalanced, accuracy tends to be deceptive.” (Ebner, 2023).

### Precision

Precision is the accuracy of the positive predictions. (Géron,2019, p.91).

### Recall

“Recall, also called sensitivity or the true positive rate (TPR): this is the ratio of positive instances that are correctly detected by the classifier.” (Géron,2019, p.91).

### F1-SCORE

“The F1 score is the harmonic mean of precision and recall. Whereas the regular mean treats all values equally, the harmonic mean gives much more weight to low values. As a result, the classifier will only get a high F1 score if both recall and precision are high.” (Géron,2019, p.92)

## Multi-Class Classification Models

In this project, I am trying to predict numbers. The MNIST dataset contains the numbers 0 **through** 9. Because the y variable has ten different classes and not **continuous** values the problem is a Multi-Class Classification problem.

“The multi-class classification … has at least two mutually exclusive class labels, where the goal is to predict to which class a given input example belongs.” ([K](https://www.datacamp.com/portfolio/keitazoumana)eita, 2022).

All the following models were tested during this project.

### Linear SVM Classification

The objective of Linear SVM Classification is to find a good balance between separating the classes (keeping the “street” as large as possible) with as few border violations as possible (data points in the street). To be able to predict accurately without overfitting. (Géron,2019, pp.153-155).

### Random Forest

“In Random Forest, each tree in the ensemble is built from a sample drawn with replacement from the training set. Furthermore, when splitting each node during the construction of a tree, the best split is found through an exhaustive search of the feature values of either all input features or a random subset of size max\_fetures.” (Scikit Learn, 2007).

### Extra-Trees Classifier

The Extra-Trees Classifier is a more random version of Random Forest. It accomplishes this by using random thresholds for each feature rather than searching for the best possible thresholds. This technique trades more bias for a lower variance. It also makes Extra-Trees much faster to train than regular Random Forests, because finding the best possible threshold for each feature at every node is one of the most time-consuming tasks of growing a tree. (Géron,2019, p.198).

### KNeigbors Classifier

“KNeighbors-based classification is a type of instance-based learning or non-generalizing learning: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.” (Scikit Learn, 2007).

### Logistic Regression

“Logistic Regression is commonly used to estimate the probability that an instance belongs to a particular class. If the estimated probability is greater than 50%, then the model predicts that the instance belongs to that class, and otherwise it predicts that it does not.” (Géron,2019, p.142).

### Voting Classifier

“The idea behind the [Voting Classifier](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html#sklearn.ensemble.VotingClassifier) is to combine conceptually different machine learning classifiers and use a majority vote or the average predicted probabilities (soft vote) to predict the class labels. Such a classifier can be useful for a set of equally well-performing models in order to balance out their individual weaknesses.” (Scikit Learn, 2007).

# Method

## Tools

Most of the programming was done using Python with sklearn (Scikit-learn) in a JupyterNotebook environment using Python 3 (ipykernel).

“Scikit-learn is an open-source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection, model evaluation, and many other utilities.” (Scikit-learn).

Some of the programming was also done using Python with both sklearn and Streamlit in Visual Studio Code in a Python 3.1 environment.

“Streamlit is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science. In just a few minutes you can build and deploy powerful data apps.” (Streamlit documentation).

## Data Collection

The data that I used was the MNIST dataset. I got the dataset from sklearns datasets using the code: mnist = fetch\_openml('mnist\_784', version=1, cache=True, as\_frame=False).

“The **MNIST** database (**Modified National Institute of Standards and Technology** database) is a large collection of handwritten digits. It has a training set of 60,000 examples and a test set of 10,000 examples. 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) which contain monochrome images of handwritten digits. The digits have been size-normalized and centered in a fixed-size image. 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.” (LeCun, Y).

## Exploratory Data Analysis (EDA)

The EDA part of the project is where you get a good base understanding of the data. It is also during the EDA that you discover how the data is structured, what types of models you might want to test, if there are any anomalies in the data that needs to be fixed, for example, some data that’s not numerical values, and so on. The MNIST dataset is very well structured and clean, so there was nothing for me to fix. The 70 000 images are divided into 10 categories with the following split:

A graph of blue bars

Description automatically generated

There is no apparent imbalance in the dataset therefore I can use the evaluation metric Accuracy Score during model selection.

## Image Transformation

In the later part of this project, I am going to predict my own handwritten numbers. For this to work I need to transform them to the same format as the MNIST dataset. To accomplish this I did the following transformations:

First, I transformed the image to (28x28) and specified interpolation=cv2.INTER\_LINEARto estimate the values of the new pixels. Then I inverted the black and white colours to make it even easier to see the number. I now use the .reshape(1,-1) to turn the 2D array into a 1D array to be able to predict the image.

The image is now transformed and able to be used, but there is still a lot of unnecessary noise around the picture. See Figure 2. To combat this, I used the code in Figure 3. What this code does is that it first specifies lower\_pixel =100 and upper\_pixel=130. Then by using a for loop it goes through every pixel in the image and checks the pixel’s colour value.

If the pixel has a lower value than 100 then it changes the value to the lowest of 0. If the pixel value is between 100 and 130 the value stays the same. And if the pixel value is above 130 the pixel gets the maximum value of 255.

I have now eliminated most of the unnecessary noise and the image is ready to be predicted. See Figure 4

A screenshot of a computer code

Description automatically generated

Figure 3 Noise removing code

A pixelated number on a black background

Description automatically generatedA number on a black background

Description automatically generated

Figure 2 Transformed image with noise Figure 4 Transformed image without noise

## Model Selection

Model Selection is an important part of any machine-learning project. It is important to choose models that are likely to have a good result.

In my case, I used a large variety of classification models. My reasoning behind that decision was that I wanted to try every model by itself, but also together as a voting classifier.

The following models were tested:

* Linear SVM Classification
* Random Forest
* Extra Trees Classifier
* KNeigbours Classifier
* Logistic Regression
* Voting Classifier (Containing all the models above)

## Model Training

This project had a somewhat short timeframe. Therefore, I anticipated there to be a problem with too long runtimes for hyperparameter grid search and model training. The way I solved this problem was by only taking a sample (about 10%) of the whole dataset and splitting it too a training set (5000) validation set (1000) and test set (1000). After when I had found the best model, I used the whole dataset with a train set (60 000) and a test set (10 000). The validation set is not needed because I don’t need to compare models.

To get the best version of every model I used GridSearchCV.

### GridSearchCV

As said by Géron(2019). “All you need to do is tell GridSearchCV which hyperparameters you want it to experiment with and what values to try out, and it will use cross-validation to evaluate all the possible combinations of hyperparameter values. Which otherwise would have to be done manually.”

#### Cross-Validation

“In cross-validation, we split our dataset into a fixed number of folds, then we run the analysis on each fold, after that we averaged the overall error estimate.” (GOYAL, 2021).

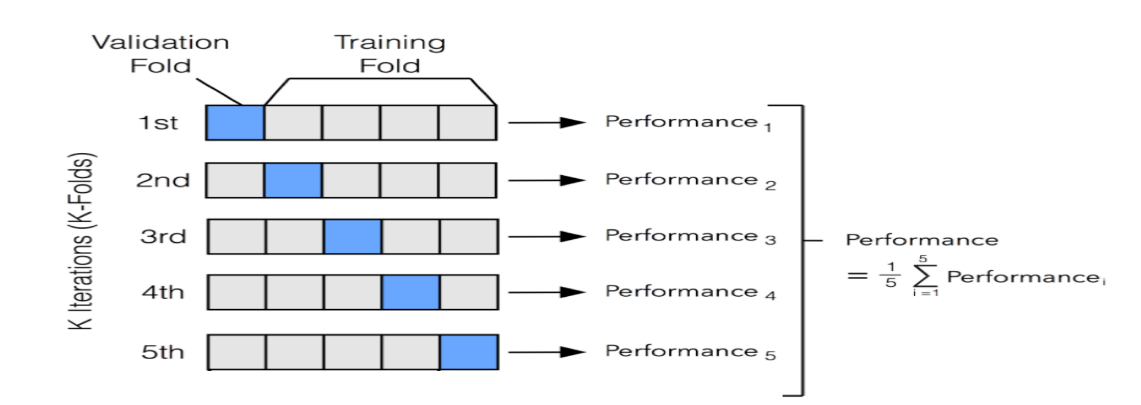


Figure 5 Cross-validation

#### Hyperparameters

These are the hyperparameters used in GridSearchCV for each tested model.

##### Linear SVM Classification

I specified the **"C”**: np.linspace(0.00001,10,15) in the grid search the rest of the hyperparameters were left on default.

The hyperparameter C determines how many border violations (data points in the street) there are. A high C value equals **fewer** border violations, and a low C value equals **more** border violations. Border violations are bad. It usually is better to have a few of them. However, a model with more border violations will probably work better on new unseen data (to generalize better). (Géron,2019, p.155).

##### Random Forest

I specified:

* **'n\_estimators'**: [50, 100, 200]
* **'max\_depth'**: [None, 10, 20, 30]
* **'min\_samples\_split'**: [2, 5, 10]
* **'min\_samples\_leaf'**: [1, 2, 4]
* **'max\_features'**: ['auto', 'sqrt', 'log2', None]
* **'bootstrap'**: [True, False]

The rest were left on default.

According to the Scikit-learn documentation:

**n\_estimators** determine the number of trees in the forest (model).

**max\_depth** determines the maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples.

**min\_samples\_split** determines the minimum number of samples required to split an internal node.

**min\_samples\_leaf** determines the minimum number of samples required to be at a leaf node.

**max\_features** determinethe number of features to consider when looking for the best split.

**Bootstrap** determines if the random sampling happens with or without replacement. If bootstrap = False, the whole dataset is used to build each tree.

##### Extra-Trees Classifier

I specified:

* **'n\_estimators'**: [50, 100, 200]
* **'min\_samples\_split'**: [1, 3, 9]
* **'min\_samples\_leaf'**: [1, 2, 4]
* **'bootstrap'**: [True, False]

The rest were left on default.

**n\_estimators, min\_samples\_split, min\_samples\_leaf, min\_samples\_leaf,** and **bootstrap** are the same as for Random Forest.

##### KNeigbors Classifier

I specified:

* **'n\_neighbors'**: [3, 5, 7, 9, 11]
* **'weights'**: ['uniform', 'distance']
* **'metric'**: ['euclidean', 'manhattan', 'minkowski']

The rest were left on default.

According to the Scikit-learn documentation:

**n\_neighbors** determines the number of neighbors to use by default for [kneighbors](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html" \l "sklearn.neighbors.KNeighborsClassifier.kneighbors" \o "sklearn.neighbors.KNeighborsClassifier.kneighbors) queries.

**Weights** determine the weight function used in prediction. Possible values:

* ‘uniform’: uniform weights. All points in each neighborhood are weighted equally.
* ‘distance’: weight points by the inverse of their distance. In this case, closer neighbors of a query point will have a greater influence than neighbors who are further away.

**Metric** determines which metric to use for distance computation.

##### Logistic Regression

I specified:

* **'C'**: [0.001, 0.01, 0.1, 1, 10, 100]
* **'penalty'**: ['l1', 'l2']
* **'solver'**: ['liblinear', 'saga']
* **'max\_iter'**: [100, 200, 300]

The rest were left on default.

According to the Scikit-learn documentation:

**C** same as for Linear SVM Classification.

**Penalty** Specify the norm of the penalty:

* 'l2' add a L2 penalty term and it is the default choice.
* 'l1': add a L1 penalty term.

**Solver** determines the Algorithm to use in the optimization problem. The default is ‘lbfgs’. To choose a solver, you might want to consider the following aspects:

* For small datasets, ‘liblinear’ is a good choice, whereas ‘sag’ and ‘saga’ are faster for large ones.
* ‘liblinear’ is limited to one-versus-rest schemes.

**Max\_iter** determines the maximum number of iterations taken for the solvers to converge.

##### Voting Classifier

I used all the models and their respective specified hyperparameters. Therefore, I didn’t do a separate grid search for the voting classifier.

## Handwritten Images

To be able to predict my own images, I used the file path for my picture and stored it in a variable with a fitting name. The next step was to transform the image to the same format as the MNIST data. To do this I take the variable that stores the file path to my image and put it through all the transformation steps mentioned before.

After doing all these steps the image is ready to be predicted. To predict the image I only have to use the following code trained\_model\_name.predict(transformed\_image), and then I can print out the number the model predicted.

## Streamlit Web App

I am going to use Streamlit to create a web app that shows the webcam live feed that predicts numbers that are held up to it.

This code below takes every single frame from the webcam, transforms that frame, and predicts what number is displayed, overlays that prediction on the not transformed image, and lastly shows that non-transformed image and the overlaying prediction in a window in the web app.

It does this very fast and to every frame, so the image and prediction in the window appear to be live.

Streamlit requires the Python script to be a continuous script and not divided into different cells like JupyterNotebook, therefore I will use Visual Studio Code for the Streamlit script.

The structure of the script is similar to predicting an image from a file in Jupyternotebook.

The Streamlit web app needs a title, so I added that. Then instead of the file path to an image I use the cap = cv2.VideoCapture(0) to store the webcam feed in a variable named cap. After that, I add some text using streamlit.write(“webcam Feed”).

I then create an empty placeholder that will be used later to display the webcam feed.

Then when that’s done, I create a (while True) function. Inside the while function I use cap.read() to save the webcam feed frame in a new variable called image. This new variable contains the frame from the webcam feed.

I took that variable and put it through the same image transformations as mentioned before, and now I have an image that I can predict using trained\_model\_name.predict(transformed\_image).

To display the prediction in the same window as the live feed I use cv2.putText() and specify inside the parentheses that I want to have the text “Prediction” to show my prediction, I also specify the location of the text, the font, and the size.

Now I have everything I need to display the live feed and the prediction. To do this I use the empty placeholder I mentioned before and put empty\_placeholder\_name.image() to specify that I want to display an image. Inside the parentheses I specify which image I want to display ((in this case) image), the colour channels of the image, and automatically adjust the width of the displayed image to ensure that the image is not cut off or stretched.

Lastly, I put cap.release() and cv2.destroyAllWindows() outside the (while True) loop. I do this because when I’m done using the program I want to stop using the webcam and I want to close the cv2 window that displays the live feed and the prediction.

# Results and Discussion

## Results

### Model Evaluation

After the GridSearchCV is done it automatically stores the hyperparameters with the best score. This makes it very easy to compare the best different models.

Because my dataset is not imbalanced, I choose to use the Accuracy Score metric to evaluate my models, instead of the earlier mentioned Precision, Recall, and F1-score. I did this because it is easy to understand and shows with what percentage the model predicts correct on, in this case, the validation data.

By using the code shown in Figure 6, I use the .score method and specify (X\_val,y\_val). By doing so I get the accuracy score of each of the best performing models on the validation data.

I also did the .score(X\_val,y\_val) on the voting classifier, the results are shown in Table 1 at the bottom of the page.

Although it was not by much the KNeigbouhrs Classifier had a slightly better score than the rest of the models.

A screenshot of a computer code

Description automatically generated Figure 6 Code example

|  |  |
| --- | --- |
| **Models** | Accuracy Score |
| Linear SVM Classification | 0.89 |
| Random Forest: Score | 0.949 |
| Extra Trees Classifier | 0.948 |
| KNeighbors Classifier | 0.95 |
| Logistic Regression | 0.899 |
| Voting Classifier | 0.946 |

Tabell 1 Accuracy score for all models

### Performance Evaluation

Now that I have my best-performing model it is time to train it on the full dataset.

I now again split the dataset into a train set and a test set. But what’s different this time is that I don’t use a validation set. That the train set size is 60 000 instead of 5 000, and the test set size is 10 000 instead of 1 000.

After I trained the KNeigbouhrs Classifier model on the new train set, I used the .score(X\_test, y\_test) to get the accuracy score on the new test set.

This time the accuracy score jumped from 0.95 on the sample dataset to 0.9717 accuracy when using the full data set.

### Handwritten Images

I tried to predict a tree of my own images, a 3, a 5, and a 2. All of them were correctly predicted, which shows that my model works on my handwritten images too. The figures show the prediction and the transformed image. See Figures 7, 8, and 9

### Streamlit Web App

My Streamlit web application also worked well and was able to correctly predict the numbers that were held up to the webcam. See the screenshots of the working web app in Figures 10, 11, and 12

A black and white image of a number

Description automatically generatedA white number on a black background

Description automatically generatedA screen shot of a number

Description automatically generated

Figure 7 Predicted 3 Figure 8 Predicted 5 Figure 9 Predicted 2

A blue number on a white surface

Description automatically generatedA screenshot of a screen capture

Description automatically generatedA blue number four on a white surface

Description automatically generated

Figure 10 Streamlit predicted 5 Figure 11 Streamlit predicted 2 5 Figure 12 Streamlit predicted 4

## Discussion

### Model

The KNeighbors Classifier worked well, but I wanted to see if the wrongly predicted numbers were evenly distributed. To see if this was the case I made a confusion matrix, Figure 13 shown at the bottom of the page. The most wrong predictions were true 4 predicted as a 9 with 19 wrongful predictions, and true 7 predicted as a 1 with 18 wrongful predictions. A handwritten 4 and 9, and a handwritten 7 and 1 can look very similar (shown in figure 14 and 15). It looks like there are only some pixels missing, so I understand why that error can occur. The images in Figures 14 and 15 are randomly selected from the MNIST datasets X. Therefore, the actual wrong predictions could look even more like the predicted number.

A chart of numbers and a number

Description automatically generated with medium confidence

Figure 13 Confusion matrix

A black and white image of symbols

Description automatically generated with medium confidenceA screenshot of a graph

Description automatically generated

Figure 14 Comparing 4 and 9 Figure 15 Comparing 7 and 1

### Handwritten Images

My KNeighbors Classifier worked well on the images that I tried to predict. One thing I learned quickly when I was first trying to get this to work was, if the image of the number that I took either had a too thin number or that the lighting was too bad. The prediction was almost always wrong.

At first, I didn’t know what was wrong but when I looked at the transformed image that was used to make the prediction, I discovered the problem.

If the lighting was too bad or the number in the image was too thin, the transformed image only looked like background noise. If the image was really bad my code bit to remove background noise turned all the pixels in the transformed image to 0. See Figure 16 for the bad not transformed image, Figure 17 for the transformed image, and Figure 18 for the transformed image with background noise elimination.

As we can see, in the transformed image it is not possible even for a human to see which number the image is supposed to contain. In the transformed image with background noise elimination, it is almost harder to make out the number.

So, it is important to have decent lighting and a somewhat thick number for an accurate prediction. As the saying goes “Shit in, Shit out”. (Prgomet, 2024).

### Streamlit Web App

The same thing as for the prediction on my handwritten number in Jupyternotebook applies to the live prediction in the web app too.

If the lighting was bad and/or the number was too thin I always got a bad prediction.

A number on a paper

Description automatically generatedA black and white image of a number

Description automatically generatedA black and white pixelated graph

Description automatically generated

Figure 16 Bad image Figure 17 Transformed bad image Figure 18 Transformed bad image with

background noise elimination

# Conclusions

When I started this project, I wanted to answer the following research questions:

1. Can I create a classification model that reaches a 95% or above accuracy score on the full dataset?
2. Can I then use that model to predict my own handwritten numbers?
3. Can I use that same model to predict my own handwritten numbers live through my webcam in a web application using Streamlit?

Answers:

1. Yes, the model that got the best accuracy score on the sample set used for grid search and model evaluation (KNeighbors Classifier) only got 95% accuracy. But when I used the full dataset (split into train and test data) to train that model I got a little over 97% accuracy when testing on the test data.
2. Yes, the model that got above 95% accuracy did correctly predict my handwritten numbers, if the handwritten numbers were not too thin and the lighting was okay.
3. Yes, I can correctly predict my handwritten numbers using my created Streamlit web app, if the handwritten numbers are not too thin and the lighting is okay.

# Teoretiska frågor

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

Tränings datan används till att träna en eller flera modeller på datan. Validerings datan används till att utvärdera sina modeller och se vilken som presterar bäst. Test datan används till att testa hur bra den bästa modellen är.

**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 ”validerings dataset”?**

Julia kan använda sig av Cross-Validation och utvärdera hur bra hennes modeller är. Efter det kan hon välja att gå vidare och testa den bästa modellen på testdatan.

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

Regressionsproblem är när den beroende variabeln y har kontinuerliga värden. Några modeller för regressionsproblem är LinearRegression, RidgeRegression, LassoRegression och PolynomalRegression.

Tillämpningsområden kan vara att prediktera en persons lön eller att prediktera vad ett hus är värt.

**4. Hur kan du tolka RMSE och vad används det till: � �𝑀𝑆𝐸 =√∑(𝑦𝑖−𝑦̂𝑖)2 𝑖**

Root Mean Squared Error (RMSE) är ett mått på hur långt det är från datapunkterna till regressionslinjen. RMSE används till att hitta en så optimal regressionslinje som möjligt. Den gör det genom att ju närmre en data punkt är till regressionslinjen desto lägre blir RMSE, låg RMSE är bra hög RMSE är dåligt.

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

Klassificieringsproblem är när den beroende variabeln y kan anta en av ett förbestämt antal klasser.

Några modeller som kan användas för klassificieringsproblem är RandomForestClassifier, KNeighborsClassifier och LinearSVC.

En Confusion Matrix är en matris som summerar hur bra en modell är. Den visar **True positives (TP), True negatives (TN), False positives (FP) och False negatives (FN). Genom dessa kan man få ut modellens recall, accuracy, precision och F1-score och där med modellens prestanda.**

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

En k-means modell är en unsupervised klustrings modell. Den används till att dela upp ett dataset i ett K antal förbestämda kluster.

Ett exempel på vad den kan tillämpas på är att dela upp en kundgrupp i mindre grupper baserat på vad dom har handlat.

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

Ordinal encoding översätter kategorier(text) till rangordnande siffror t.ex Liverpool till 1, Luton Town till 2 och Manchester United till 3.

One-hot encoding översätter kategorier(text) till icke rangordnande siffror t.ex grön till [0.0.1], röd till [0.1.0] och blå till [1.0.0].

Dummy variable encoding används oftast till regressionsproblem och översätter kategorier(text) till icke rangordnande siffror men gör det lite annorlunda jämfört med One-hot encoding, t.ex grön till [0.1], röd till [0.0] och blå till (**inte** [0.1] eller [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 har rätt då det är upp till en själv och modellens behov om man anser att datan ska vara ordinal eller nominal. Det finns inga exakta regler, därför har Göran fel.

**9. Kolla följande video om Streamlit: https://www.youtube.com/watch?v=ggDa RzPP7A&list=PLgzaMbMPEHEx9Als3F3sKKXexWnyEKH45&index=12 Och besvara följande fråga: - Vad är Streamlit för något och vad kan det användas till?**

Streamlit är ett open-source pyton bibliotek och det används för att snabbt och relativt enkelt omvandla data scripts för att skapa webapplikationer för maskinlärning och data science projekt.

# Självutvärdering

1. Utmaningen i början var att det tog väldigt lång tid att köra gridsearch för alla modeller. Jag löste problemet genom att använda mig av en mindre sample av datan för gridsearch delen.

Ett annat problem var att få Streamlit att fungera. Jag löste det problemet genom att först skapa ett helt fungerande live predikterings program i jupyternotebook. Och först efter det lägga in koden i visual studio och göra några relativt små anpassningar för att få streamlit webappen att fungera.

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

Jag anser att jag ska ha betyget VG då jag uppfyller kraven:

För betyget Väl Godkänd ska den studerande:

* + Uppnått kraven för betyget Godkänd
  + I en skriftlig rapport lösa ett problem genom att implementera metoder och modeller från maskininlärning på ett fördjupat sätt med hög säkerhet.
  + Redogöra för och kritiskt diskutera modellval, modellanpassning och modellutvärdering

1. Något du vill lyfta fram till Antonio?

Jag tycker att både kursens utbildningsdel och kunskapskontroll var väldigt bra.

Det var extra roligt att se att jag faktiskt kunde genomföra ett helt ML projekt från början till slut själv.

Mycket rolig kurs!

# References

[IBM Data and AI Team](https://www.ibm.com/blog/author/ibm-data-and-ai/) (2024, February 1). *10 everyday machine learning use cases*. IBM Blog. https://www.ibm.com/blog/10-everyday-machine-learning-use-cases/

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

Keita, Z. (2022, February). *Classification in Machine Learning: An Introduction*. Datacamp. https://www.datacamp.com/blog/classification-machine-learning

*1.11.2.1. Random Forests*. Scikit Learn. (2007). https://scikit-learn.org/stable/modules/ensemble.html#forest

*1.6.2. Nearest Neighbors Classification*. Scikit Learn. (2007). https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors-classification

*1.11.4. Voting Classifier*. Scikit Learn. (2007). https://scikit-learn.org/stable/modules/ensemble.html#voting-classifier

*Getting Started*. Scikit Learn. (2007). https://scikit-learn.org/stable/getting\_started.html

*Streamlit documentation*. Streamlit. (n.d.). https://docs.streamlit.io/

LeCun, Y. (n.d.). THE MNIST DATABASE of handwritten digits. http://yann.lecun.com/exdb/mnist/

Ebner, J. (2023, November 16). *Classification Accuracy, Explained*. Sharp Sight. https://www.sharpsightlabs.com/blog/classification-accuracy-explained/

*Confusion Matrix in Machine Learning*. GeeksforGeeks. (2024, March 6). https://www.geeksforgeeks.org/confusion-matrix-machine-learning/

GOYAL, C. (2021, May 21). *Importance of Cross Validation: Are Evaluation Metrics enough?*. Analythics Vidhya. https://www.analyticsvidhya.com/blog/2021/05/importance-of-cross-validation-are-evaluation-metrics-enough/