HAN_UNIVERSITY
OF APPLIED SCIENCES

Machine Learning report

HAN Embedded Vision and Machine Learning

[WOUTER PEETERS WEEM, 610002]

[BART VAN DEELEN, 627366]

[JASPER HULSTEIN, 627055]

GROUP: [GROUP 9]

DATE: [30-11-2021]

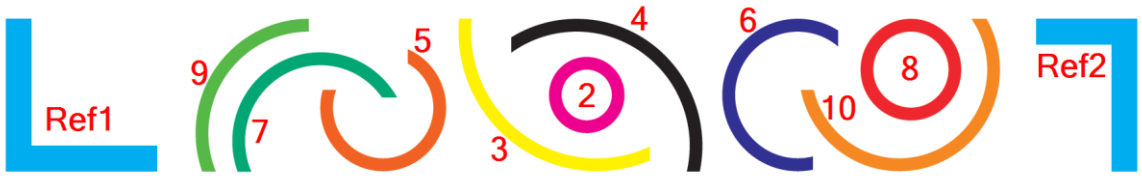
By submitting this portfolio, the authors certify that this is their original work, and they have cited all the referenced materials properly.

Contents

1 INTRODUCTION

Voor het vak Embedded Vision Design 3 van de deeltijd hoofdmodule Embedded Vision and Machine Learning moet een mini-project gemaakt worden waarin het bouwen van een Machine Learning applicatie centraal staat. Dit portfolio beschrijft het Machine Learning (ML) mini-project van onderwerp tot conclusie. ML is in de brede zin erg relevant in veel nieuwe ontwikkelingen waar informatie uit een beeld of video gehaald moet worden. Een voor de hand liggend voorbeeld hiervan is bijvoorbeeld het herkennen van verkeersborden in een zelfsturende auto.

Het miniproject dat wij gaan maken is onderdeel van het overkoepelende EVML-project. Voor dit project gaan wij de register merktekens die op de verschillende drukwerken van een Flexo drukmachine van MPS-systems gedrukt worden analyseren om de afwijking van de merktekens ten opzichte van elkaar te bepalen. De merktekens worden naast elkaar op het substraat gedrukt met een referentie merkteken aan beide kanten die door het eerst drukwerk gedrukt worden. Hierdoor ontstaat een merken patroon met maximaal 10 verschillende drukmerken. Figuur 1 laat een complete merktekenset met deze drukmerken zien:



Figuur 1-1 Merkteken set (Uitvergroot van 22,5 x 3 mm)

Het doel van het project is om de positie van elk merkteken ten opzichte van de referentie merktekens separaat weer te kunnen geven aan de machine. Om dit te bereiken is het eerst van belang dat de merktekens geclassificeerd worden volgens de voorgeschreven nummering zoals in figuur 1. Wanneer dit bekend is kan de positie van elk merkteken bepaald worden ten opzichte van de referentie merktekens en doorgegeven worden aan de machine.

Het classificeren van de merktekens willen we met ML doen. Door dit met ML te doen is het mogelijk om kleine afwijkingen in de merktekens zoals een te kort gedrukte lijn, of een te dik gedrukte lijn te trainen. Aangezien er 10 verschillende klassen zijn is het erg lastig om dit met voorwaarden te coderen. Ook is het, wanneer er een aangepaste merktekenset gebruikt zou worden, niet nodig om alle voorwaarden opnieuw te schrijven, maar kan het maken van foto's en het trainen van het model genoeg zijn.

Onze leerdoelen in dit project zijn erg breed aangezien we nog geen ervaring met ML of Embedded Vision hebben. Bij dit miniproject is het voornamelijk van belang dat we leren omgaan met alle onderwerpen die komen kijken bij het bouwen van een ML toepassing. Dit zijn onder andere data acquisitie, onderzoek en voorbereiding, ML model selectie, training en finetunen en het in gebruik nemen en testen van het model.

2 PROBLEM STATEMENT

Het hoofddoel van de Machine Learning (ML) die wij toe gaan passen is het herkennen van de verschillende merktekens binnen een set. Globaal gezien moet van een merkteken set een foto gemaakt worden, waarna de afzonderlijke merktekens herkend worden. Dit resultaat wordt in het overkoepelende EVML-project vervolgens gebruikt om de merktekens te lokaliseren.

Voor dit project wordt ervan uitgegaan dat merktekens nooit wijzigen. Ze mogen wel verplaatsen binnen een set, maar de vorm van een merkteken moet op elke set nagenoeg gelijk zijn. Indien de vormen toch wijzigen, moet het systeem opnieuw ingesteld worden.

Het systeem voor dit project moet gebouwd worden volgens de traditionele stappen van het vision image processing. Dit geeft een aantal voordelen.

- Het project heeft een structuur die internationaal geaccepteerd is.
- De structuur van het project is inzichtelijk. Op die manier is een probleem makkelijk toe te wijzen aan een bepaalde stap in het proces.

De overige functionele eisen van het systeem zijn weergegeven in **Error! Reference source not found.**.

Tag	Omschrijving	Prioriteit
FR01	De data voor het model bestaat uit foto's van merkteken sets. Deze sets bevatten allen dezelfde type merktekens. De formaten en vormen komen dus overeen.	1
FR02	De foto's zijn gemaakt met een gelijkwaardige setup. Dezelfde camera en gelijke belichting.	1
FR03	De kwaliteit van de foto's is gelijkwaardig.	1
FR04	Er zijn voldoende foto's om een 2 sets te maken. 1 set voor training, 1 set voor testen.	1
FR05	De foto's moeten voorbewerkt worden, om alle merktekens te kunnen herkennen.	1
FR06	Van alle merktekens in de training set moeten de features worden opgeslagen.	1
FR07	Een machine learning model moet de opgeslagen features gebruiken om zichzelf te trainen.	1
FR08	Het model moet getest worden, met een minimumresultaat van 95%. Er mag nooit een verkeerd merkteken gekozen worden. Dit kan aanleiden tot onterechte aansturing van de machine.	1
FR09	Het complete systeem moet getest worden tijdens. Hierbij moet het systeem voldoende feedback en inzicht geven, om tot een goede conclusie te komen.	2

Figuur 2-1 Functionele eisen

3 DATA ACQUISITION AND EXPLORATION

3.1 Verzamelen beelden

Het belangrijkste onderdeel van een Machine Learning project is de dataset. De uiteindelijke nauwkeurigheid van ons systeem zal voor het grootste deel afhankelijk zijn van de kwaliteit van onze dataset, veel meer als van de kwaliteit of keuze voor een specifiek algoritme.

Om een goede dataset te kunnen vergaren zullen we dus een acquisitie systeem op moeten zetten waarmee we op een eenvoudige manier snel veel data kunnen genereren van uiteraard goede kwaliteit. Met een dataset van goede kwaliteit bedoelen we data die:

- Goed gebalanceerd is, ieder merkteken komt evenveel voor in de dataset waardoor er geen voorkeur kan ontstaan voor een bepaald merkteken.
- Compleet is, er zijn weinig tot geen datapunten die bijvoorbeeld belangrijke informatie over features missen of waarbij deze informatie onjuist is.
- Schoon is, eventuele outliers zijn verwijderd of niet meegenomen in de dataset.
- Voldoende beschikbaar is, er is voldoende data aanwezig om een model te kunnen trainen.

Voor ons project zullen we een dataset moeten gaan genereren van data die we uit diverse beelden van merktekens kunnen halen. We hebben er in dit geval voor gekozen om al deze beelden te maken met een vaste opstelling waardoor we zo min mogelijk variatie hebben in de beelden. Denk hierbij aan de afstand tot het merkteken, type camera, type belichting etc. De opstelling bestaand uit een Raspberry pi met pi camera.

Figuur 3-2 Opstelling pi

Figuur 3-1 Opstelling pi

Door middel van deze gecontroleerde opstelling hebben we een groot aantal beelden kunnen maken van verschillende merktekens. Om ons systeem uiteindelijk te kunnen beoordelen op zijn nauwkeurigheid hebben we een deel van deze beelden direct aan de kant gezet zodat we op het eind kunnen testen met een set beelden die ons model nog nooit gezien heeft.

3.2 Data acquisitie

Nu we voldoende beelden verzameld hebben zullen we uit ieder beeld de merktekens moeten herkennen, labelen en de features verzamelen. In eerste instantie halen we alle features uit het beeld die we kunnen bedenken. Later gaan we kijken welke features juist wel of niet geschikt zijn. Kleur is bijvoorbeeld een feature die niet geschikt is aangezien iedere pers verschillende soorten kleuren kan drukken. Het merkteken van een bepaalde klasse kan dus het ene moment rood zijn en het volgende moment groen.

Features die wellicht wel geschikt zouden kunnen zijn:

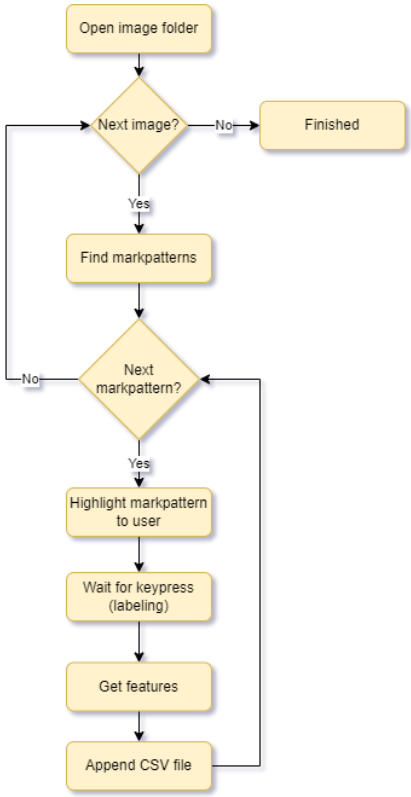
- Omtrek
- Oppervlakte
- Radius
- Oppervlakte convex hull
- Hoek (richting) open kant van boog
- Radius minimum enclosing circle
- Oppervlakte minimum enclosing circle
- Verschil tussen oppervlakte merkteken en oppervlakte enclosing circle
- Aantal hoekpunten
- Afstand diepste punt tot convex hull

Het python script dat dit voor ons doet gaat als volgt te werk.

In een loop gaan we langs alle beelden in de map met trainingsdata. Op iedere afbeelding voeren we een aantal operatoren uit om ruis te verwijderen en om uiteindelijk de merktekens te kunnen herkennen.

Als alle 11 de merktekens zijn gevonden gaan we weer in een loop langs ieder merkteken en berekenen we hier alle features. In beeld wordt nu het huidige merkteken aangemerkt zodat de gebruiker via het toetsenbord kan aangeven welk merkteken dit is zodat het juiste label en de gevonden features kan worden gekoppeld.

Dit doen we voor ieder merkteken in ieder beeld en zo vullen we een CSV bestand. Ieder gevonden merkteken voegt een regel toe met het juiste label en alle gevonden features.



Figuur 3-3 Flowchart aquire_data.py

	A	B	C	D	E	F	G	H
1	Press code	Angle	Distance t	Perimeter	Area	IsConvex	Corners	MinEnclosingRadius
2	Press 2	7.596.375.653.207.350	406	5.996.711.345.911.020	25544.0	FALSE	15	9.140.494.537.353.510
3	Press 5	6.718.034.430.201.720	50815	12.796.265.382.766.700	23020.0	FALSE	16	1.435.823.211.669.920

Figuur 3-4 Example features.csv

```python
# General method to get all mark patterns in image
def get_mark_patterns(img, noise_filter_size):
    contours, hier = cv2.findContours(img, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    mark_patterns = []
    for contour in contours:
        area = cv2.contourArea(contour)
        if area > noise_filter_size:
            mark_patterns.append(MarkPattern(contour))

    return mark_patterns
```

Figuur 3-7 Python script om alle markpatterns te vinden in een plaatje

```python
class MarkPattern:
    # Constructor method for markPattern
    def __init__(self, img):
        self.Contour = img
        self.StartPoint = None
        self.EndPoint = None
        self.FarthestPoint = None
        self.MidPoint = None
        self.prediction = None
        self.position_percentage = None
        self.Perimeter = cv2.arcLength(self.Contour, True)
        self.Area = cv2.contourArea(self.Contour)
        self.IsConvex = cv2.isContourConvex(self.Contour)
        self.Hull = cv2.convexHull(self.Contour, returnPoints=False)
```

Figuur 3-6 Python class die een markpattern representeert

```python
# Get press code from key press
def get_press_code(x):
    print(x)
    if x == 49:  # 1
        return "Press 9"
    if x == 50:  # 2
        return "Press 7"
    if x == 51:  # 3
        return "Press 5"
```

Figuur 3-5 Python method om een label te verkrijgen door een toets in te drukken

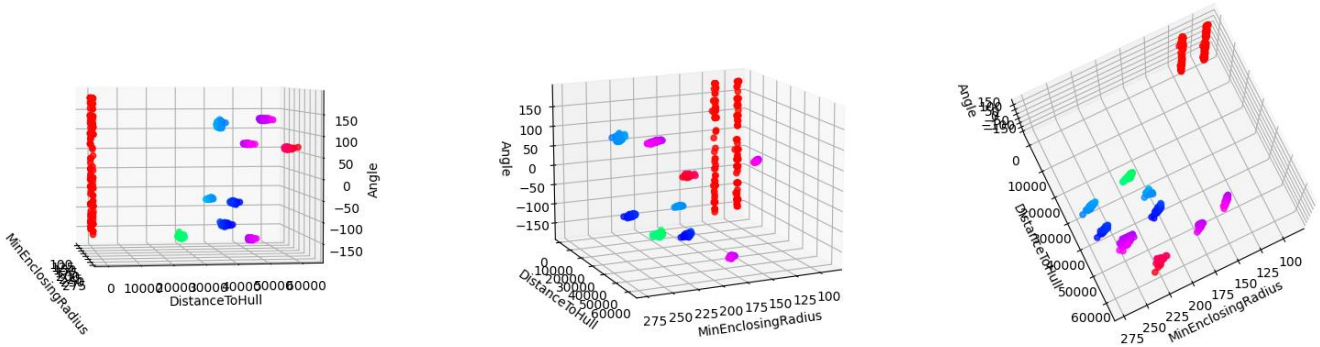
3.3 Analyseren data

Deze dataset kunnen we nu beter gaan analyseren, welke features zijn goed en welke juist niet?

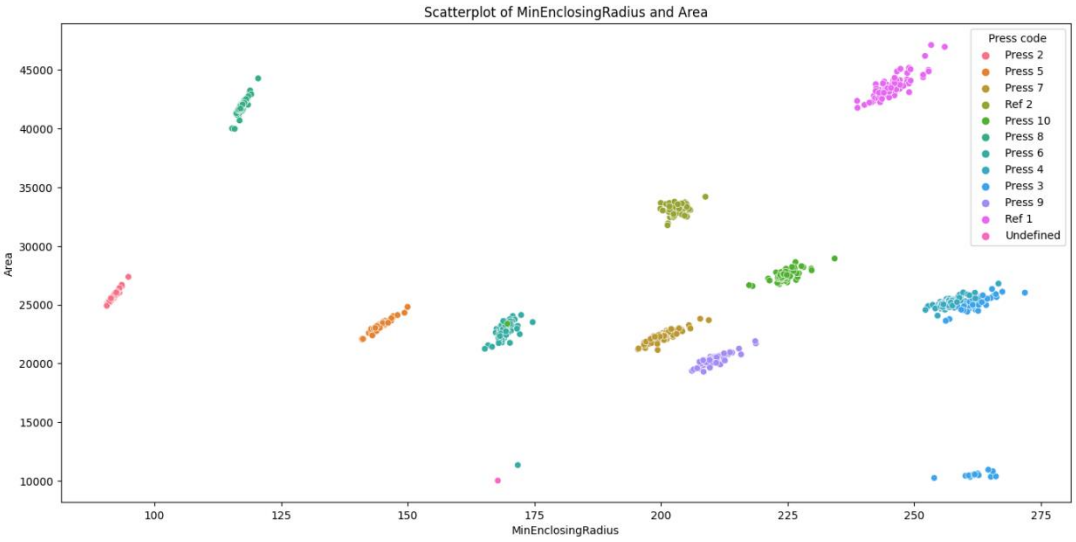
Als we het hebben over een goede feature bedoelen we dat deze feature de volgende eigenschappen heeft.

- De feature is informatief, nuttige informatie over het merkteken
- Hij is discriminerend, de feature vertelt dus echt iets over wat dit merkteken anders maakt dan andere merktekens
- Hij is onafhankelijk, dit betekent dat de feature op zichzelf staat en niet afhankelijk is van een andere feature. Zo zou oppervlakte bijvoorbeeld afhankelijk kunnen zijn van lengte.
- De feature is uniek en voegt dus echt iets aan informatie toe wat andere features niet hebben.

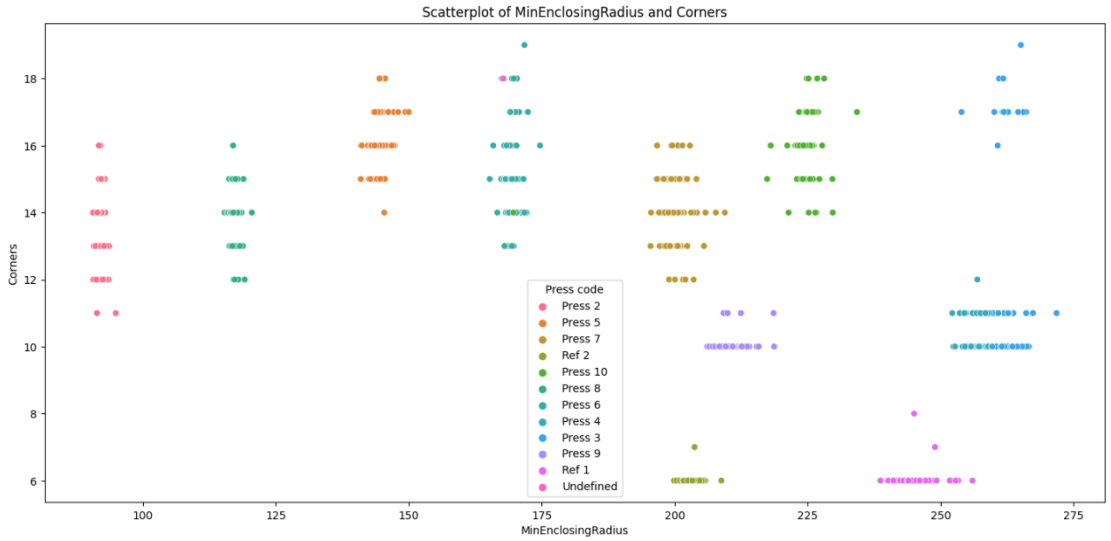
Om een beter beeld te krijgen van onze data hebben we deze eerst op diverse manieren gevisualiseerd.



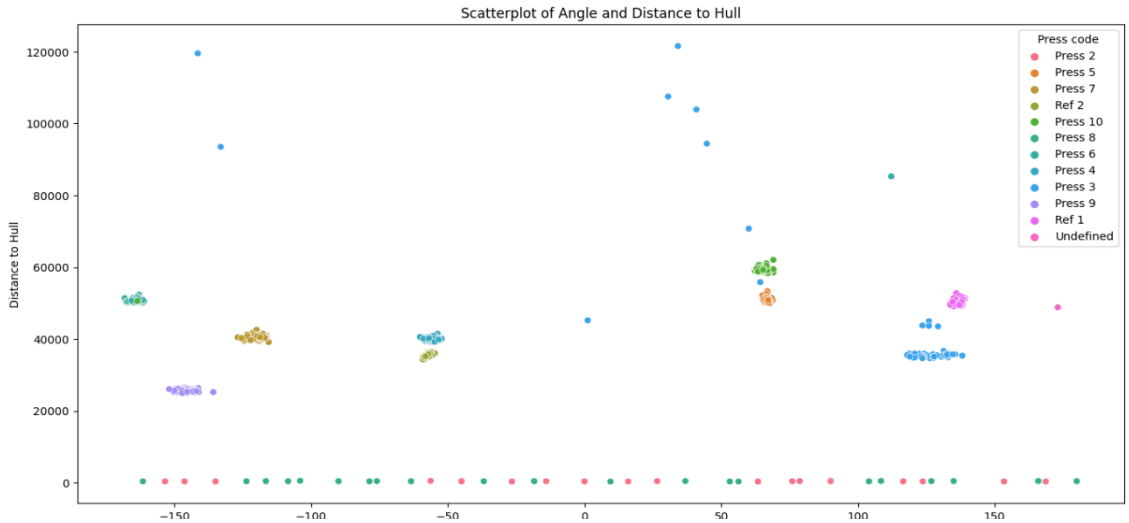
Figuur 3-8 3d scatterplot of Angle, DistanceHull, MinEnclosingRadius



Figuur 3-9 Scatterplot van de MinEnclosingRadius vs de oppervlakte



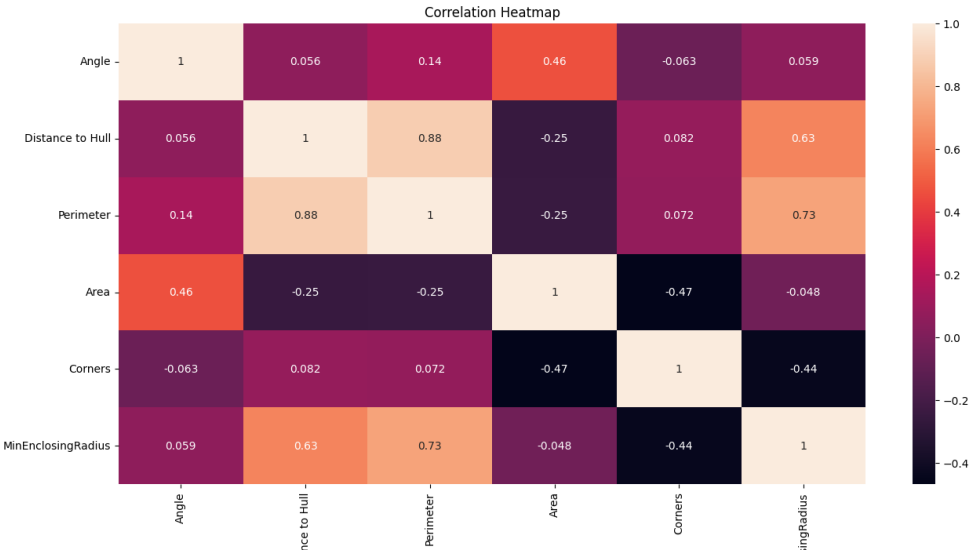
Figuur 3-10 Scatterplot van het aantal hoeken vs de minEnclosingRadius



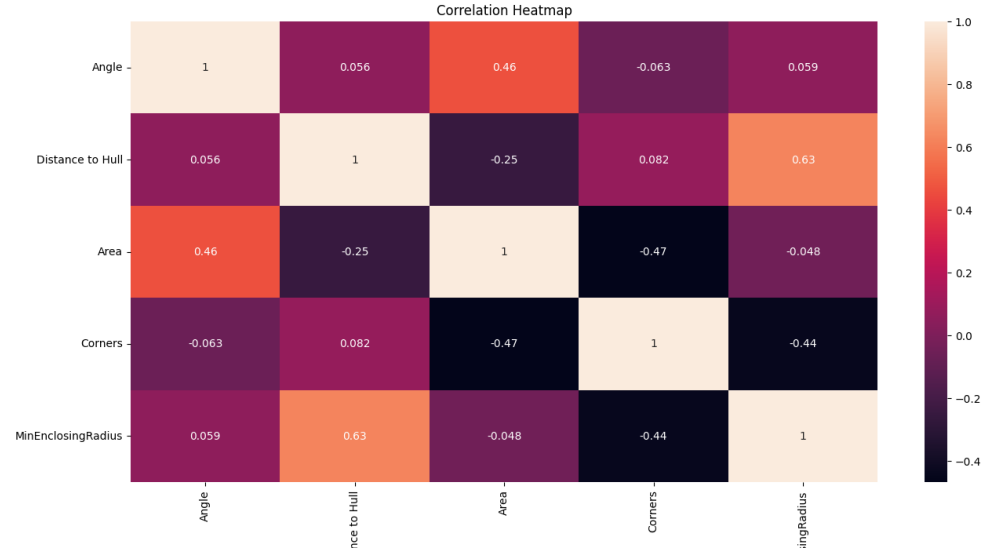
Figuur 3-11 Scatterplot van de Angle vs DistanceToHull

Zoals bovenstaand te zien is geven de scatterplots een goed beeld van onze data. Door een aantal features te combineren zouden we een duidelijk onderscheid tussen de verschillende klassen moeten kunnen maken.

Om inzicht te krijgen in welke features eventueel geschikt zijn om te combineren hebben we correlation heatmaps gemaakt. Deze plots geven aan hoe gecorreleerd de verschillende features zijn.



Figuur 3-13 Correlation heatmap van alle features



Figuur 3-12 Correlation heatmap van de gekozen features

Na het bestuderen van deze heatmaps hebben we gekozen de Omtrek als feature uit de dataset te laten om dat deze erg hoog gecorreleerd is met de DistanceToHul. Dit is ook te verklaren aangezien de DistanceToHull in veel gevallen de radius van de vorm benaderd.

4 MODEL SELECTION, TRAINING AND VALIDATION

4.1 Classifier kiezen

Nu we een goed overzicht hebben over onze data moeten we een classifier gaan kiezen.

Er zijn tientallen ML algoritmes beschikbaar en zullen hier een keuze in moeten maken. In ons project is een duidelijk onderscheid te maken tussen verschillende klassen die geïdentificeerd moeten worden, namelijk 9 merktekens en 2 referentietekens. Deze 11 tekens hebben allen identieke kenmerken waardoor ze van elkaar te onderscheiden zouden moeten zijn.

Omdat we na de acquisitie en het cleanen van de data een mooie dataset hebben met features en gelabelde outputs (supervised learning) kunnen we de selectie toepassen op een aantal veelgebruikte classifiers.

- K Nearest Neighbour
- Decision Tree
- Random Forest
- SVM

We hebben gekozen om te starten met een decision tree aangezien onze merktekens dusdanig eenvoudig te onderscheiden waren dat we de classifiers ook zelf zouden kunnen coderen. Een decision tree leek ons dan de eenvoudigste oplossing en later het best te verklaren. Waarom kiest het algoritme op een bepaald moment voor de een en niet voor de ander.

Met de decision tree classifier hebben we in het begin veel testen gedaan en een proof of concept voor onszelf bewezen. De classifier gaf ons op dat punt al behoorlijk goede resultaten. Voor het echt goed cleanen en analyseren van onze schaarse data op dat punt zat de nauwkeurigheid vaak al boven de 95%.

Later zijn we ook andere algoritmes beter gaan testen en moesten we concluderen dat de KNN classifier toch echt betere resultaten gaf dan de decision tree. Na het goed cleanen van de data konden we stellen dat ons model een nauwkeurigheid van 100% haalt. Ook de precison en de recall zijn 1.0 met de KNN classifier.

4.2 Splitten data

De totale dataset bestaat uit 100 verschillende foto's van merktekens. Van deze 100 hebben we direct 15 beelden aan de kant geschoven om de laatste test mee uit te voeren. Zo weten we zeker dat het algoritme deze beelden nog nooit gezien heeft.

Met de andere 85 beelden gaan we het algoritme trainen en een cross-validation uitvoeren om te zien hoe goed ons model presteert.

De cross validation split onze data steeds in 5 sets, eentje voor validatie en 4 voor training. Het model wordt dus getraind met 4/5 van de set en gevalideerd met 1/5 van de set. Dit herhalen we dan 10 keer om 10 keer de nauwkeurigheid van het model te bepalen. Het gemiddelde resultaat van deze 10 metingen is dan onze uiteindelijke nauwkeurigheid.

Het testen van de verschillende algoritmes op steeds op dezelfde wijze gedaan.

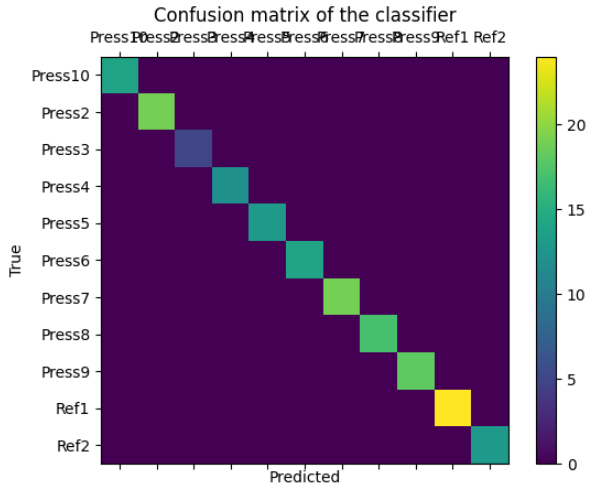
1. Inlezen dataset
2. Standaardiseren data
3. Cross-validatie uitvoeren op data

```
# Read complete dataset from csv
data = pd.read_csv('C:\\Users\\woute\\OneDrive - HYTORC Nederland
BV\\Documenten\\School\\ML\\evml-2021-print-mark-
recognition\\src\\data\\features\\markPattern_features_cleaned.csv')

# Remove unused features
X = data.drop(columns=['Press code', 'IsConvex', 'Perimeter'])

# y are only the labeld results
y = data['Press code']

# Scale the data
scalar = StandardScaler()
scaled_X = scalar.fit_transform(X)

# Create a classifier
myClassifier = KNeighborsClassifier(n_neighbors=3)       # 0.996 (0.004) -->
1.000 (0.000)

# Evaluate the model
cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=10, random_state=36851234)
scores = cross_val_score(myClassifier, scaled_X, y, scoring='accuracy', cv=cv)
print('Cross Accuracy: %.3f (%.3f)' % (mean(scores), std(scores)))
```

4.3 Evalueren model

Ons model presteert eigenlijk zo goed dat optimaliseren lastig is. De bottleneck in ons systeem is het preprocessen van de data. Als de beelden goed binnenkomen en ieder merkteken gevonden wordt classificeert het model altijd goed.

Buiten dat het belangrijk is om ervoor te zorgen dat we zo vaak mogelijk het goede merkteken herkennen is het nog belangrijker om ervoor te zorgen dat ons model in ieder geval nooit het verkeerde merkteken herkend. Dit zou namelijk betekenen dat er een afwijking wordt gemeten die er helemaal niet is. De precisie voor iedere klasse moet dus 100% zijn.



Figuur 4-1 Confusion matrix

	precision	recall	f1-score	support
Press 10	1.00	1.00	1.00	15
Press 2	1.00	1.00	1.00	15
Press 3	1.00	1.00	1.00	9
Press 4	1.00	1.00	1.00	15
Press 5	1.00	1.00	1.00	19
Press 6	1.00	1.00	1.00	11
Press 7	1.00	1.00	1.00	13
Press 8	1.00	1.00	1.00	18
Press 9	1.00	1.00	1.00	18
Ref 1	1.00	1.00	1.00	16
Ref 2	1.00	1.00	1.00	19
accuracy			1.00	168
macro avg	1.00	1.00	1.00	168
weighted avg	1.00	1.00	1.00	168

Figuur 4-2 Classificatie rapport

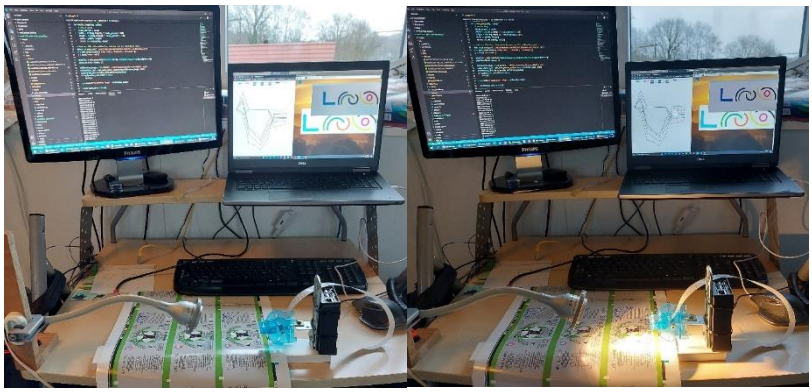
5 DEPLOY AND TEST

Nadat alle testen zijn afgerond en ons model naar tevredenheid presteert gaan we het systeem draaien op een Raspberry pi met een pi-camera.

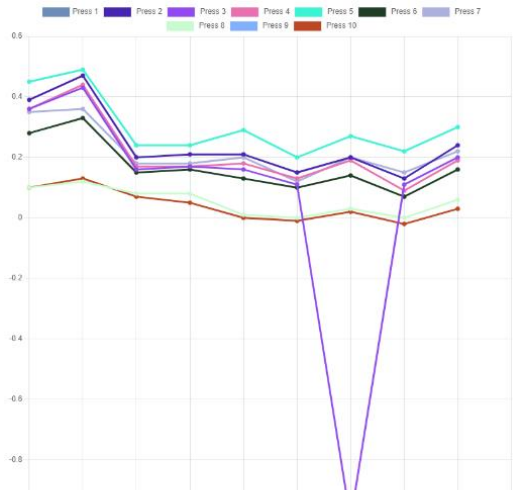
Met deze hardware kunnen we het systeem simuleren zoals het bij de klant zou kunnen werken. Een bedrukte baan met merktekens komen onder de camera voorbij en via een extern trigger signaal wordt er opdracht gegeven aan de pi-camera om een foto van het merkteken te schieten.

Het beeld wordt hierna verbeterd en gesegmenteerd waarna ieder merkteken met zijn berekende features aan het model wordt aangeboden om te voorspellen op welke plek welk merkteken zich bevindt. Nu zijn we in staat om voor ieder merkteken de positie en eventuele afwijking ten opzichte van de referentietekens te berekenen.

Deze test voeren we uit met de 15 beelden die het model nog nooit gezien heeft waardoor we kunnen zien of het algoritme goed generaliseerd.



Figuur 5-1 Test opstelling raspberry-pi



Figuur 5-2 Grafiek met gemeten afwijkingen

6 CONCLUSION

Het Machine Learning gedeelte van ons project is meer dan geslaagd.

We zijn in staat om telkens ieder merkteken uit het beeld correct te classificeren. Het was erg interessant en leerzaam om zoveel inzicht in een dataset te krijgen en deze vervolgens zo te preparen dat je hier een model mee kunt trainen.

Door middel van bijvoorbeeld de diverse scatterplots en feature distribution plots vonden we dat onze preprocessing pipeline nog wel wat te wensen overliet, deze hebben we hierna kunnen verbeteren.

Onder aan de streep is het herkennen van onze merktekens met Machine Learning wellicht een klein beetje overkill omdat de tekens zulke duidelijke en simpele kenmerken hadden. Dit hadden we ook handmatig kunnen programmeren met een aantal if statements. Dat wil niet zeggen dat ML geen goede oplossing is, we hebben door ML toe te passen wel veel inzicht gekregen in het hoe en wat van het herkennen van de merktekens.

7 REFERENCES

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow.* Sebastopol, Canada.: O'Reilly Media.

SMART criteria. (2020, 05 14). Opgehaald van wikipedia: https://en.wikipedia.org/wiki/SMART_criteria

CODE APPENDICES