DAKI 4 – Kingdomino P0 Projekt

(https://github.com/jonas0711/DAKI4-P0.git)

Gruppens navne:

Peter Gravgaard

Rasmus Mølby

Jonas Holm Ingvorsen

Nanna Krogh Jensen

Mikkel Stouby Holm

Stoyan Dimitrov Mihaylov

Vilde Mork-Knudsen

Vejleders navn:

Lasse Riis Østergaard

Indholdsfortegnelse

1. Problemanalyse

Hvorfor er det smart at have et program, der kan tælle point automatisk?
For personer med kognitive skader eller hjerneskade, især de der husker livet før deres skade, er selvstændighed en afgørende del af rehabiliteringen. Mange har svært ved at multitaske, og kan kun have fokus på 1 ting ad gangen. Opgaver som at spille et spil og samtidig holde styr på pointene kan være udforende. Teknologiske hjælpemidler kan derfor hjælpe disse personer med at genvinde en oplevelse/følelse af kontrol samt selvstændighed. Et program, der automatisk tæller point i spil, kan lette denne opgave, øge livskvaliteten og mindske afhængighed af menneskelig støtte.

1.2 Om Kingdomino

Kingdomino kategoriseres som et strategisk familiebrætspil (8+), hvor spillere bygger kongeriger ved at placere todelt terrænbeklædt "domino-lignende"-brikker. Målet er at score flest point ved at skabe sammenhængende terræner med kroner. Spillet er kendt for at være hurtigt at lære og tager ikke mere end 15-20 min. Vinderen er den spiller med flest point. Spillet blev lanceret i 2017, af den franske brætspilsdesigner, Bruno Cathala.

1.2.1 Kingdominos spilleregler

- 2 spillere: 24 dominobrikker i spil, 2 konger/spillere, varighed 6. runder
- 3-4 spillere: 36(3) - 48(4) brikker, 1 konge/spiller, varighed 12. Runder

Hver dominobrik har to terræner, som skal placeres korrekt i spillernes kongerige. Et kongerige består af 5x5 felter.
Terræntyper: Der er syv terræntyper i spillet: Mark, Skov, Sø, Græs, Sump, Mine og "Hjem" felt.
Hver hjemfelt har en tilhørende kongebrik i samme farve. Inden start, placeres hjemfeltet, hvor kongeriget bygges ud fra, Hjemfeltet kan være sammenhængende med alle terræntyper, men giver ikke point.
Point: Point scores ved at gange antal sammenhængende terræn-felter af samme type ganget med antal kroner forekommet på det givet terræntypes terrænfelter.

Eksempel: Lad os sige, 4 felter af ens og sammenhængende terræntype, men det givne terræn ingen kongekroner indeholder, giver terrænet altså, 0 point. Har spilleren derimod to brikker af samme terræn, med to kroner på, får man så 4 point (2x2).

1.2.2 Hvad adskiller de forskellige terræntyper fra hinanden?

Der er i alt 7 forskellige terræntyper i Kingdomino. De har hver især deres unikke kendetegn som gør det muligt for spillerne, samt et evt. Program, at genkende de individuelle terræntyper.

Figur 1: Kingdominos terrænbrikker med navn. Dette billede viser de forskellige terræntyper, samt hvad de heder. Dersom en brik har en, to eller tre kongekroner på, får den tilsvarende tal efter terræntypenavnet.

Helt grundlæggende kan man adskille terræntyperne ud fra forskellige parameter. Udover deres visuelle udtryk har forskellige farver samt temaer, er visse spilmæssigt svære at sammensætte. Nogle terræntyper giver generelt højere point, f.eks. Mine-terræn. Mine-terræn fremkommer blot 6 gange i hele spillet, men 5/6 felter indeholder kongekroner, dermed, formår en spiller at sammensætte et mine-terræn er spilleren garanteret point. Til sammenligning fremgår terrænet "Field" i alt 26 gange, hvor 5/26 indeholder kun én kongekrone. Field kan risikere at optage en del plads, uden at hente point hjem. De 7 forskellige terræntyper, er ikke blot forskellige visuelt, men er også forskellige i sandsynlighed indenfor point og hyppighed.

1.2.3 Hvordan adskiller de forskellige terræntyper visuelt fra hinanden?

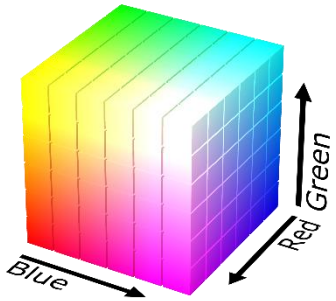
Terræntyperne er generelt meget forskellige, rent farvemæssigt og designmæssigt. Nogle brikker indeholder elementer i kontrastfulde farver, såsom et gult bådhus i feltet "Lake1" (se Figur 1), som står i kontrast med terrænets blå farve. For os mennesker er det logisk, at "Lake1" er et terrænfelt tilhørende kategorien "Lake", for et program, som udgangspunkt, nok ikke. Derfor, vil der arbejdes meget med farvespektrum på de individuelle brikker/terræntyper. Her er det umiddelbart nemmest at finde systematiske forskelle, som gør det muligt at adskille terræntyperne.

1.3 Om digitale billeder

Da vores datasæt til denne opgave består udelukkende af digitale billeder, har vi vurderet det relevant af formidle en grundlæggende forståelse for hvad et digitalt billede egentligt er, samt hvordan det kan bruges som data.

1.3.1 Hvad er digitale billeder?

Digitale billeder består af data, der vises som farvekoder gennem "Pixels" på en skærm. Farverne i et billede dannes oftest via RGB-system, hvor (R)ed, (G)reen og (B)lue kombineres til forskellige farver. Hvert billede består af tusindvis af pixels, som tilsammen skaber det endelige billede.



Kilde:
https://upload.wikimedia.org/wikipedia/commons/a/af/RGB_color_solid_cube.png

1.3.2 Dataudtræk fra billeder

For at udtrække data fra billeder bruges billedanalyseprogrammer, der kan genkende farvemønstre. I dette projekt vil der udvikles på et program, der kan analysere farverne på et samlet Kingdomino kongerige, deraf genkende terrænerne og deraf, beregne point baseret på farveforskellene i de forskellige terræner.

1.4 Endelig problemformulering & afgrænsning

Hvordan kan man designe et billedanalysesystem, som skal anvendes til automatisk at beregne point i brætspillet "Kingdomino".

1.5 Kravspecifikation

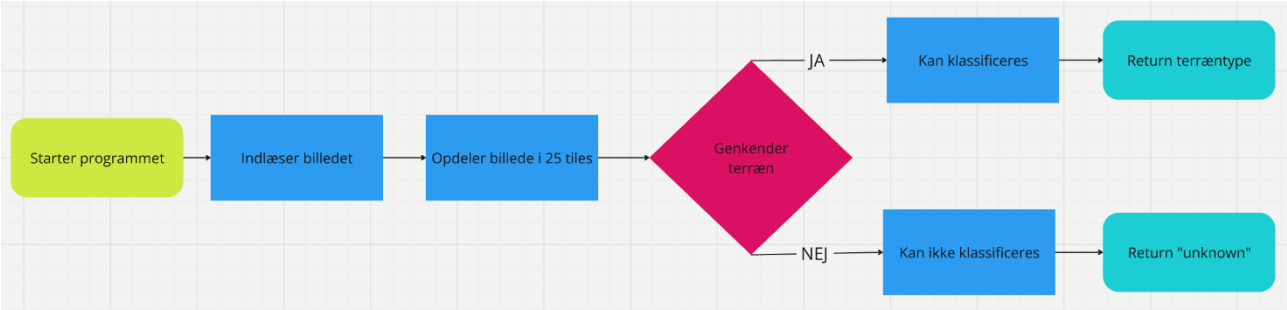
Vi vil udvikle et program der automatisk kan beregne scoren i brætspillet Kingdomino. Vi udvikler programmet med kodesproget Python i IDE-programmet Visual Studio Code. Vores program skal kunne klassificere og skelne mellem de forskellige terræn-typer herunder: Grassland, Field, Forest, Mine, Lake, Swamp og Home. Denne skelnen mellem terræn-typerne er essentiel for vores programs beregning af point scoren, da point scoren jo bl.a. afhænger af grupperinger af brikker af samme terræn-type. Vores program skal altså vide om det er samme- eller forskelligartede brikker der ligger grupperet. Vi inddrager dog *ikke* at vores program skal kunne skelne mellem terræntyper med og uden kongekroner.

Programmet skal kunne benyttes af enhver der spiller spillet, det skal være simpelt og ligetil at starte op, og der er naturligvis ingen forudsætning for at man har kendskab til kodning for at anvende programmet. Som nævnt tidligere i vores indledende problemformulering, har vores program til formål at hjælpe med optælling og overblik over den endelige point score i Kingdomino med særligt fokus på at give støtte til kognitivt skadede mennesker. Vores program skal altså være så simpelt som muligt at bruge.

2. Metode

Dette afsnit redegør for arbejdsprocessens fremgangsmåder og de bagliggende teknikker, der er anvendt ift. At løse projektets problemstilling samt opnå det ønskede resultat. Fremgangsmåde bag arbejdsprocessen redegøres i overskuelighed, hvilket gør det muligt at evaluere metodevalgene. Afhængig af relevans, vil der yderligere blive redegjort for ledende overvejelser bag metodevalg og eventuelle begrænsninger, der kan have påvirket projektets resultat.

For at vores program kan opfylde vores krav, har vi skulle løse mange små problemer. Her kommer forklaringer for de valg vi har taget, samt deres betydning for vores endelige program.



Figur 2: Flowchart-visualisering af processen for vores program.

2.1 Pre-processering

Det første vores program skal kunne, er at indlæse billeder af brætspillet, samt opdele billederne i såkaldte “tiles”. Dette fik vi opnået meget hurtigt, da det var inkluderet i kode-skabelonen som vi fik uddelt ved projektets start. Det er denne kode-skabelon som danner fundamentet for vores endelige program, da både fil-indlæsningen, “tile”-opdeling og HSV-klassificeringen er noget vi genbruger, fordi det fungerer.

2.2 Klassifikation

I den oprindelige kode-skabelon benyttes funktionen “np.(mean)”, som beregner gennemsnittet af HSV-værdierne (Hue, Saturation, Value). Tidligt i projektet besluttede vi dog at anvende medianen i stedet for gennemsnittet til vores værdier. Vi oplevede, at måden den oprindelige kode-skabelon kategoriserede de forskellige terræntyper, ikke var tilfredsstillende. Vi undersøgte om kode-skabelonen kunne optimeres på en nem og effektiv måde. På trods af en del fejlselektioner var der tilstrækkelig med gode data, hvilket førte til overvejelsen om at bruge median frem for gennemsnit.

Median sortere HSV-værdierne i stigende rækkefølge og fokusere på de værdier, der forekommer i midten. Medianen finder altså, midten af midten, af en givet data, ved at tage de to midterste værdier, lægge dem sammen og dernæst dividere med to. Median fremstod hurtigt som en fordelagtig løsning, da median ikke påvirkes af lave eller høje værdier, som gennemsnittet gør.

Vi betragtede forskellene mellem de to metoder, og konkluderede hurtigt, at brugen af medianen ville være mere robust ift. ekstreme værdier. Gennemsnittet kan skævvride resultatet, især i Kingdomino hvor både meget høje og meget lave point kan opnås. Derudover, kan dårlig belysning og billedkvalitet også føre til ekstreme værdier, særlig inden for S-værdi (Saturation). Dette forstærkede yderligere vores valg af median fremfor gennemsnit.

2.3 Databehandling

En af de større udfordringer var at aflæse data direkte fra terminalen. Det var tidskrævende at afskrive data manuelt fra terminalen over i et Excel-dokument og skabte mange uundgåelige tastefejl. For at afhjælpe dette, udviklede vi, i samarbejde med ChatGPT, en forlængelse af vores kode til at analysere og finde HSV-værdierne for alle billeder i vores dataset, og derefter indsætte dem i et Excel-ark. Excel-arket indeholder altså en oversigt over hver tile med dens tilhørende HSV-værdier for hvert billede. Herefter bruger vi ChatGPT igen til at formulere et stykke kode, som indsætter alle billeder med en oversigt over tiles og de tilhørende ‘True labels’. Nu har vi altså to Excel-filer med henholdsvis tiles med HSV-værdier, og tiles med ‘True labels’. Vi “merger” nu disse Excel-filer til én samlet Excel-fil med oversigt over hver tile med dens HSV-værdier og ‘True label’. Sammenlægningen af disse data i én fil gjorde, at vi nemmere kunne visualisere sammenhængen mellem de prædikterede værdier og de faktiske terræntyper.

Senere, viste Excel-formatet sig at være til fordel, det gjorde det let at integrere yderligere data og skabe en confusion-matrix, som gav et konkret indblik i programmets succesrate, præcision og hvor fejlklassifikationerne nøjagtigt forekom.

2.4 Resultatbehandling

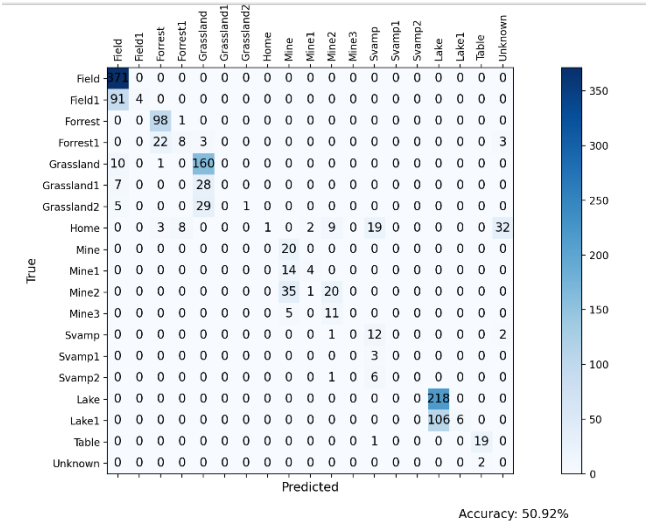
En confusion-matrix er et værktøj der kan bruges til at lave tabeller der sammenligner de prædikterede og sande værdier af din data. Den kan anvendes til at identificere hvor godt en klassifikationsmodel fungerer. Det er en god og simpel måde at få et overblik over, hvilke forudsigelser der er rigtig og forkerte, samt hvilke fejl modellen laver. En confusion-matrix viser os hvordan de faktiske terræntyper sammenlignes med de forudsagte terræntyper.

Vi har valgt at visualisere vores confusion matrix ved hjælp af kode, så det blev nemmere at forudse hvad der rent faktisk sker i vores kode. Dette gør det muligt for os at identificere hvor det går godt og hvor det går galt, altså hvilke terrænbrikker der bliver korrekt og forkert identificeret af vores program.

En af de udfordringer vi stødte på, var at vores home-brikker ofte blev forkert klassificeret i vores confusion-matrix. Homebrikkerne har fire forskellige farver, samt er der nogle gange et slot på, som tit overlapper med farverne på terræntyperne. Dette trak vores nøjagtighed (accuracy) ned. Som en løsning på dette problem eksperimenteret vi med at fjerne homebrikken fra vores kode, og lave en ny confusion-matrix uden home. Dette medførte os en meget bedre nøjagtighed, hvilket gav os et godt grundlag til at se væk fra homebrikken videre i projektet. Vi har altså brugt vores confusion-matrix ikke kun til at evaluere hvor godt vores model fungerer, men også til at identificere eventuelle brikker der skaber problemer.

3. Test

I vores testfase har vi gennemført tre forskellige testforsøg for at evaluere programmets evne til at klassificere terræntyperne i Kingdomino. Testene blev udført med henblik på at forbedre klassifikationen ved gradvist at justere og optimere vores tilgang. Confusion-matrix blev anvendt til at visualisere programmets præstation og identificere områder med fejlklassifikationer. Nedenfor beskrives de tre testfaser, samt en samlet evaluering.



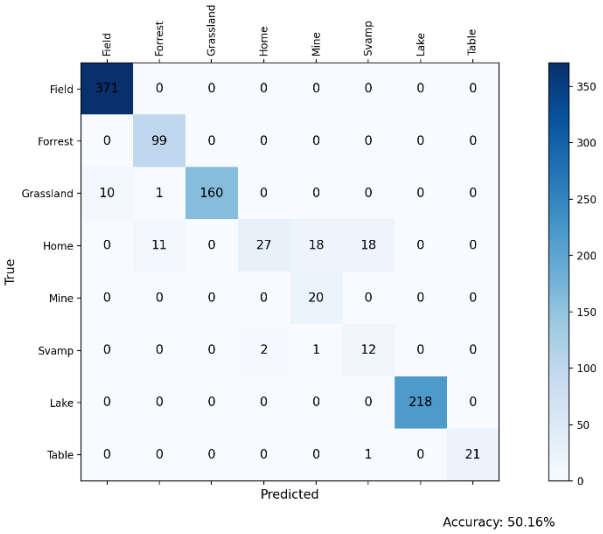
Figur Test 1.1

1. Første test: Indeksering af terræner og kroner

I den første test indekserede vi hver terræntype baseret på antallet af kroner for at skabe en mere detaljeret klassificering (f.eks. "Lake1" for en Lake med én krone).

Resultater:

- **Klassifikationsnøjagtighed:** 50,92%
- **"Field":** 371 korrekte klassifikationer
- **"Lake":** 218 korrekte klassifikationer
- **"Lake1":** 6 korrekte klassifikationer, 106 fejlagtige klassifikationer som "Lake"
- **"Home":** 27 korrekte klassifikationer, 11 fejlagtige som "Forest" og 18 fejlagtige som "Mine"
- **"Grassland":** 160 korrekte klassifikationer, fejlagtige klassifikationer som "Grassland1" og "Grassland2"



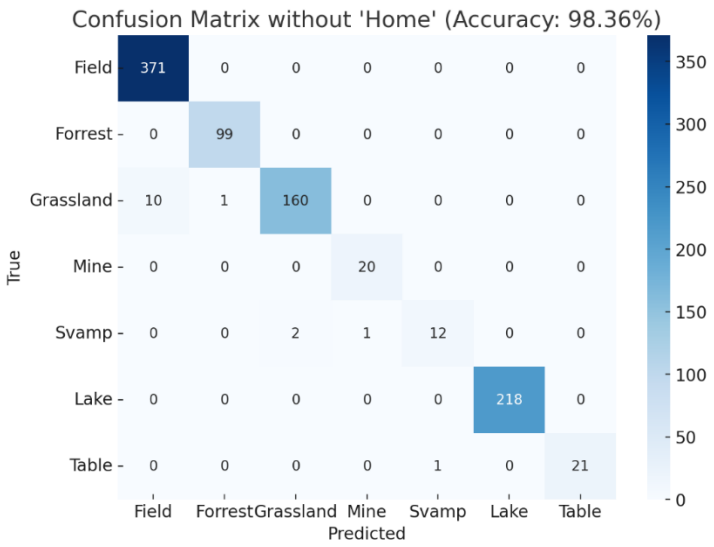
Figur Test 2.1

2. Anden test: Forenkling af terræntyper

Efter den første test forenklede vi klassifikationen ved kun at fokusere på de overordnede terræntyper uden at inkludere antallet af kroner.

Resultater:

- **Klassifikationsnøjagtighed:** 50,16%
- **"Field":** 371 korrekte klassifikationer
- **"Lake":** 218 korrekte klassifikationer
- **"Grassland":** 160 korrekte klassifikationer, 10 fejlagtige som "Field" og 1 fejlagtig som "Forest"
- **"Home":** 27 korrekte klassifikationer, 11 fejlagtige som "Forest" og 18 fejlagtige som "Mine"
- **"Swamp":** 12 korrekte klassifikationer, enkelte fejlagtige klassifikationer



Figur Test 3.1

3. Tredje test: Fjernelse af "Home"-brikken

I den tredje test ekskluderede vi "Home"-brikken fra datasættet for at undersøge dens indflydelse på klassifikationsnøjagtigheden.

Resultater:

- **Klassifikationsnøjagtighed:** 98,36%
- **"Field":** 371 korrekte klassifikationer
- **"Lake":** 218 korrekte klassifikationer
- **"Grassland":** 160 korrekte klassifikationer, 10 fejlagtige som "Field" og 1 fejlagtig som "Forest"
- **"Mine":** 20 korrekte klassifikationer
- **"Swamp":** 12 korrekte klassifikationer

3.1 Samlet evaluering

I den samlede testproces har vi gennemført tre testforsøg med henblik på at evaluere og optimere programmets evne til at klassificere terræntyperne i Kingdomino. Ved at justere klassifikationsmetoden og ekskludere bestemte terræntyper har vi observeret ændringer i klassifikationsnøjagtigheden. Confusion-matrix har været et nyttigt værktøj til at identificere specifikke områder, hvor programmet har præsteret godt, samt områder med udfordringer.

4. Evaluering af projektstyring og brug af ChatGPT

I begyndelsen af hele projektet startede vi processen med en god plan og struktur for det kommende arbejde, men da den kreative tankeproces blev igangsat, blev projektstyringen meget flyvsk. Set i bakspejlet, har vi for tidligt taget fat i testfasen. Jo dybere vi begik os i testning, jo mindre overblik fik vi over metode-delen, da vi ikke fik rapporteret vores vej derhen imens vi var i processen. Det har været udfordrende at skrive metodeafsnittet, når vi i fællesskab har måtte tænke tilbage for at finde hoved og hale i vores faktiske metode.

Herudover har forskellige niveauer i kodning forårsaget en opdeling af arbejdsopgaver således at kodningen blev udført af de med tidligere erfaring. Dette er selvfølgelig svært at undgå i en forholdsvis stor gruppe med forskellige udgangspunkter ift. kodningsevne, men vigtigt at tage med til fremtidige projekter at alle får fingrene på tastaturet med kodningen af projektet.

I vores projekt har ChatGPT hjulpet med flere vigtige opgaver, især inden for kodning og dataanalyse. Vi brugte den til at generere Python-kode til Excel-integration, så vi nemt kunne udtrække data og gemme det i Excel-filer. Den assisterede også med at opsætte en confusion matrix, der hjalp os med at evaluere programmets præcision ved klassificering af terrænerne. Derudover har vi brugt ChatGPT til stavekontrol og idé-generering, hvor den gav forslag til alternative metoder og hjalp med at lave simulering af dem. Selvom ChatGPT var nyttig, krævede det en grundlæggende viden om Python for effektivt at kunne anvende den genererede kode. Desuden kunne ChatGPT give forkerte oplysninger og hallucinere, hvilket betød, at vi altid skulle faktatjekke resultaterne. Samlet set var ChatGPT et værdifuldt redskab i projektet, der hjalp os med at automatisere processer og forbedre vores løsninger. Dog var det nødvendigt med baggrundsviden og kritisk tænkning for at sikre præcision i vores resultater.

5. Diskussion

Selvom vores confusion-matrix og den samlede Excel dataanalyse har været til stor hjælp med at optimere på programmets præcision, så er det meget vigtigt at belyse, at programmet stadig er sårbart over for forskellige faktorer. Billedbelysning, billedkvalitet og identifikation af homebrikken er problemstillinger der hidtil ikke kunne løses. Selv med medianberegning, som reducerede indflydelsen af ekstreme værdier, forekom der stadig udfordringer i sådan en grad, at programmets funktionalitet i

praksis vil være betvivlende. Lysforhold er en typisk udfordring i billedanalyse og uundgåeligt i de fleste tilfælde. Vi har under processen diskuteret mange løsningsforslag, for at optimere programmet og løse udfordringerne. Der har været løsningsforslag indenfor andre billedanalysemetoder ift. at designe programmet til f.eks. at kunne identificere kronerne, adskille brikkerne fra hinanden efter form/struktur og genkende mønstre/former m.m. Disse metoder omhandler bl.a. Machine-learning og viden der er udenfor vores kompetencer, og ligger på et niveau vi ikke helt kan omfavne (endnu). Der er blevet diskuteret og ideudviklet på et evt. "flag-system" baseret på varians beregning. Tanken var, at programmet skulle identificere de felter med for stor varians, de der ville blive fejlkategoriseret, og derefter automatisk inddele disse felter i yderligere felter og fjerne yderkanten, og identificere feltet på ny, men udelukkende trække data ud fra midten af feltet. Dette ville fjerne kronerne placeret ved feltets kant, fjerne slottet fra homebrikkens overlapning af et andet felt, og undgå at betragte en evt. Bordplade, fordi brikken er en smule skævt placeret. Det er bare et af flere flag-systemer vi har diskuteret. Havde vi kunne formå at komme i mål med den videreudvikling, ville vi forvente, en større accuracy i confusion-matrix med homebrikken inkluderet.

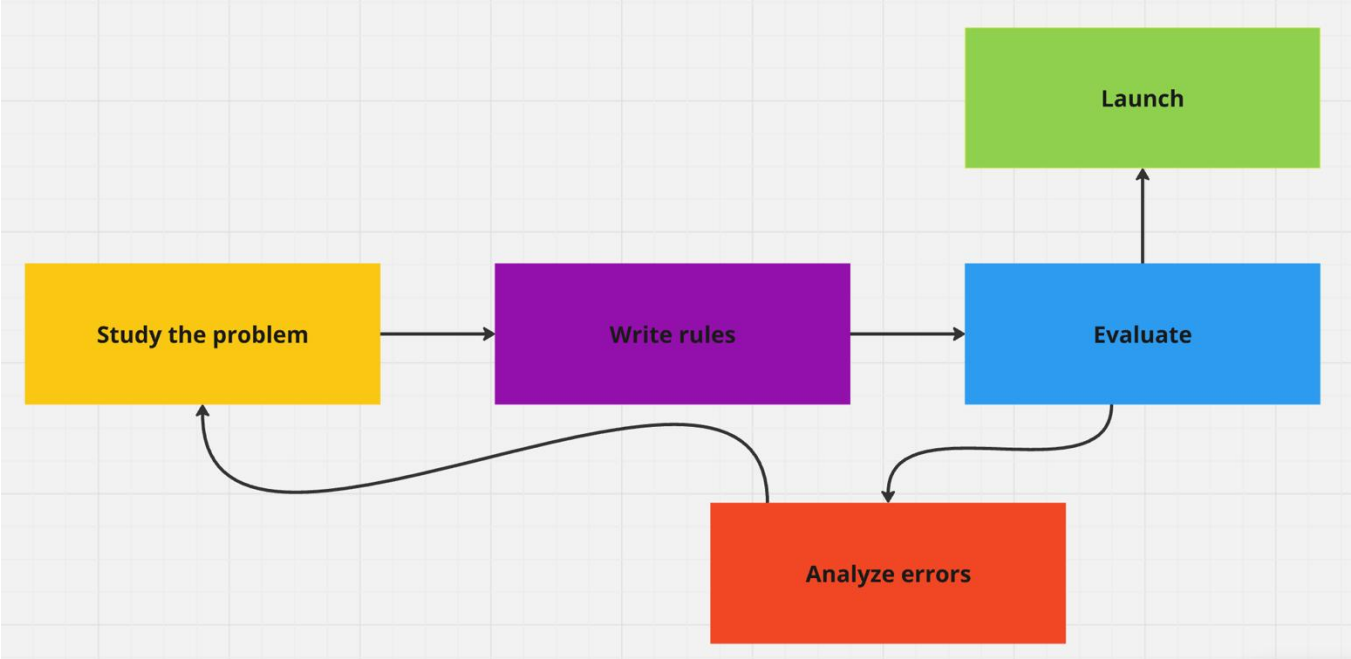
6. Konklusion

I dette projekt har vi udviklet en model der ved hjælp af billedanalyse og farvegenkendelse kan identificere de forskellige terræntyper i brætspillet Kingdomino. Vi har i projektet brugt et datasæt af 74 billeder og har udført tre forskellige test, hvor hver test er udført med hele datasættet. Vi har altså ikke brugt Machine Learning i projektet. I den første test klassificerede vi hver terræntype baseret på antallet af kroner for at skabe en mere detaljeret klassificering f.eks. "Lake1" for en Lake med én krone. Dette gav en accuracy score på 50.92%. I anden test klassificerede vi kun de overordnede terræntyper og inkluderede ikke antallet af kroner. Dette gav en accuracy score på 50.16%. I tredje test klassificerede vi igen kun de overordnede terræntyper, ligesom i anden test, men uden at inddrage 'Home' brikken. Dette gav os en accuracy score på 98.36%. Dette projekt har altså givet forskellige resultater afhængigt af hvad der inddrages i klassificeringen af terræntyperne. I den kode-skabelon vi har fået til projektet, skal alle terræntyper naturligvis bruges og vi må derfor tage udgangspunkt i vores anden test, der inddrager alle overordnede terræntyper inklusive Home. HSV-intervallerne brugt i denne test og den tilhørende accuracy på 50.16% er altså vores endelige resultat.

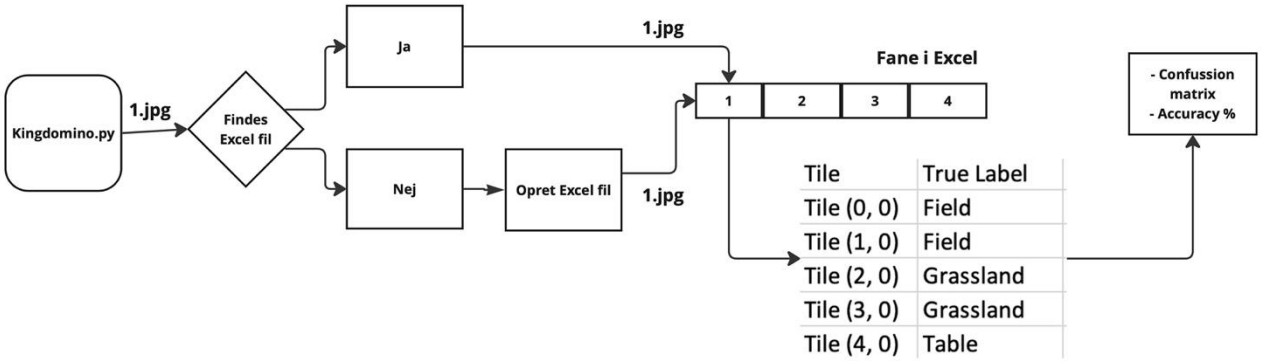
7. Bilag

```python
import cv2 as cv
import numpy as np
import os

# Main function containing the backbone of the program
def main():
    print("+-------------------------------+")
    print("| King Domino points calculator |")
    print("+-------------------------------+")
    image_path = r"C:\Users\admin\Downloads\King Domino dataset\1.jpg"
    if not os.path.isfile(image_path):
        print("Image not found")
        return
    image = cv.imread(image_path)
    tiles = get_tiles(image)
    print(len(tiles))
    for y, row in enumerate(tiles):
        for x, tile in enumerate(row):
            print(f"Tile ({x}, {y}):")
            print(get_terrain(tile))
            print("=====")

# Break a board into tiles
def get_tiles(image):
    tiles = []
    for y in range(5):
        tiles.append([])
        for x in range(5):
            tiles[-1].append(image[y*100:(y+1)*100, x*100:(x+1)*100])
    return tiles

# Determine the type of terrain in a tile
def get_terrain(tile):
    hsv_tile = cv.cvtColor(tile, cv.COLOR_BGR2HSV)
    hue, saturation, value = np.mean(hsv_tile, axis=(0,1)) # Consider using median instead of mean
    print(f"H: {hue}, S: {saturation}, V: {value}")
    if 0 < hue < 0 and 0 < saturation < 0 and 0 < value < 0:
        return "Field"
    if 0 < hue < 0 and 0 < saturation < 0 and 0 < value < 0:
        return "Forest"
    if 0 < hue < 0 and 0 < saturation < 0 and 0 < value < 0:
        return "Lake"
    if 0 < hue < 0 and 0 < saturation < 0 and 0 < value < 0:
        return "Grassland"
    if 0 < hue < 0 and 0 < saturation < 0 and 0 < value < 0:
        return "Swamp"
    if 0 < hue < 0 and 0 < saturation < 0 and 0 < value < 0:
        return "Mine"
    if 0 < hue < 0 and 0 < saturation < 0 and 0 < value < 0:
        return "Home"
    return "Unknown"

if __name__ == "__main__":
    main()
```

Bilag 1: Kode-skabelon



Bilag 2: The traditional approach



Bilag 3: Brug af Excel-filer i vores proces