CHALMERS

GÖTEBORGS UNIVERSITET

Maskininlärning för att analysera biotiska och abiotiska faktorer i algodling

Machine learning to analyze biotic and abiotic factors in cultivation of algae

Kandidatarbete inom civilingenjörsutbildningen vid Chalmers

Adam Bernstad
Filip Björnåsen
Idun Pontusdotter
Hussein Rezai

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2025

Maskininlärning för att analysera biotiska och abiotiska faktorer i algodling

Kandidatarbete i matematik inom civilingenjörsprogrammet Bioteknik vid Chalmers
Adam Bernstad
Idun Pontusdotter

Kandidatarbete i matematik inom civilingenjörsprogrammet Teknisk fysik vid Chalmers
Hussein Rezai

Kandidatarbete i matematik inom civilingenjörsprogrammet Teknisk matematik vid Chalmers
Filip Björnåsen

Handledare: Marija Cvijovic

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2025

Förord

Detta arbete är en produkt av ett samarbete mellan fyra individer från tre olika program på Chalmers tekniska högskola som förts samman av intresset för att använda maskininlärning för att modellera och optimera biologiska system. På grund av våra skilda tidigare akademiska erfarenheter hade vi vid projektets början också olika kunskaper och kompetenser. Adam och Idun med inriktningen Bioteknik hade inledningsvis en större förståelse för kiselalgers biologi och vilka faktorer som möjligtvis kan påverka deras tillväxt. Filip och Hussein, med bakgrund i Teknisk matematik respektive Teknisk fysik, hade istället högre kompetens inom programmering, databehandling och modellering, vilket visade sig vara mer relevant för arbetet.

Vi valde att inte dela upp de olika aspekterna av arbetet lika tydligt som vissa andra grupper, detta eftersom vi alla ville lära oss mer om maskininlärning, men såklart fick varje gruppmedlem ändå olika specifika uppgifter efter hand. Undantaget var Idun som redan från början fick större ansvar för texten i rapporten än resterande gruppmedlemmar, i utbyte mot ett mindre ansvar för databehandling samt modellträning och anpassning.

I den inledande litteraturstudien fick olika gruppmedlemmar ansvar att läsa på om olika maskininlärningsmodeller. Tanken var också att varje gruppmedlem själv skulle träna den typ av modell de läst upp sig på, men detta visade sig vara omöjligt på grund av den begränsade mängden data. Senare inom det tekniska arbetet fick Filip och Hussein uppgiften att visualisera och hitta korrelationer inom data, samtidigt som Adam fick ansvar för hyperparameteroptimering, men alla tre hjälptes åt med databehandling och för att träna XGBoost-modeller på olika parameterkombinationer. Under arbetets gång fördes en loggbok för att dokumentera gruppmedlemmarnas specifika bidrag och tidsinvestering.

Gruppmedlemmarnas bidrag till rapportskrivningen ges i tabellen nedan.

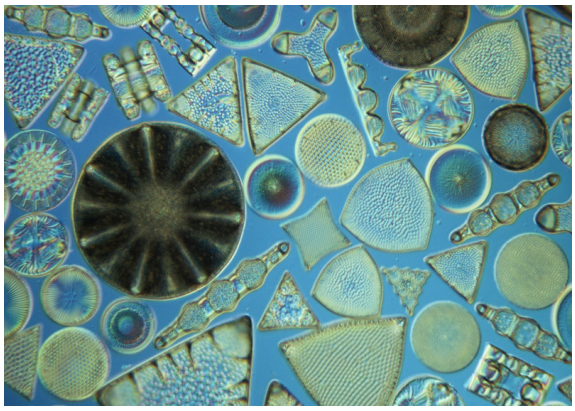
Gruppmedlem	Avsnitt
Adam Bernstad	Förord, 3.3, 3.4, 3.5, 4.3.4, 5.2
Filip Björnåsen	Sammandrag, 2.4, 3.1, 3.2, 4.1, 5.3, 5.4, 5.5
Idun Pontusdotter	Populärvetenskaplig presentation, 1.1, 2.1, 2.2, 2.5, 4.2
Hussein Rezai	1, 1.2, 2.3, 4.3.1-3, 5.1

Slutligen vill vi tacka vår handledare Marija Cvijovic som har varit delaktig i varje steg av arbetet och hjälpt oss på vägen. Vi vill också tacka Swedish Algae Factory för att de delat sin data med oss och möjliggjort det här arbetet.

Populärvetenskaplig presentation

Har du alger i din hudkräm? Det kan låta osannolikt, men det finns många användningsområden för kiselalger, och hudvård är ett av dem.

Kiselalger är små organismer som står för ungefär 20% av syreproduktionen på jorden. Som man hör på namnet består kiselalger bland annat av kisel, i själva verket har de ett skal gjort av kiseldioxid. Det här hårda yttre skalet är det som ger kiselalger deras speciella egenskaper. I skalet finns mönster av porer, som skiljer sig mellan arter (se Figur 1).



Figur 1: En mikroskopbild på olika arter av kiselalger. De små prickarna som syns inuti algerna är porerna [1].

Inom hudvård kan kiselalger ha flera funktioner. Algica, som produceras av Swedish Algae Factory, består av kiselalger och används som en rengörande och fuktighetsbevarande ingrediens. Dessutom kan det skydda huden mot luftföroreningar och öka solskyddsfaktorn i din solkräm.

Det finns många fler saker man kan använda kiselalger till förutom hudvård. Man kan göra biodiesel, och djurfoder som kan ersätta majs och soja. Kiselalger kan också användas inom läkemedel, eftersom de har bevisad effekt mot inflammation och cancer. De kan till och med användas för att öka effektiviteten hos solpaneler.

Trots alla möjligheter för användning av kiselalger, är produktionen idag låg. Hur kan man öka produktionen, så att vi kan dra nytta av algernas egenskaper? Genom att analysera olika faktorer såsom pH, temperatur och ljus kan vi förbättra odlingsförhållandena, och därmed öka skörden. Att analysera en stor mängd data för hand är svårt och tidskrävande, men det blir genast lättare om man använder maskininlärning.

Maskininlärning är ett sätt att träna en dator att lösa en uppgift som den inte är programmerad för. Då kan man med hjälp av datorn se samband mellan odlingsförhållandena och skörden, och på så sätt få veta om t.ex. en högre temperatur ger en större skörd. Det går också att ta reda på vilka faktorer som har ingen eller negativ påverkan på skörden. Med hjälp av resultaten kan sedan odlingen anpassas, och skörden ökas.

Förmodligen har du ännu inga kiselalger i din hudkräm, inte heller i din diesel. Men förhoppningsvis kan vi i framtiden se hur kiselalgerna tillämpas i vår vardag, till djurfoder och solpaneler tack vare den ökade skörden. Möjligtvis kan också den här metoden att analysera odlingsförhållandena appliceras på andra grödor, så att jordens resurser kan användas på det bästa möjliga sättet.

Sammandrag

Maskininlärning har utvecklats till en central teknik med breda tillämpningsområden, särskilt i takt med att tillgången till stora datamängder och beräkningskapaciteten ökat. Denna utveckling har möjliggjort mer kraftfulla modeller för att analysera komplexa system och göra prediktioner.

Syftet med denna studie är att med hjälp av maskininlärning undersöka biotiska och abiotiska faktorer som påverkar odling av kiselalger, med särskilt fokus på att förstå tillväxten och minska kontaminering av grönalger i odlingssystemet. Datainsamlingen är gjord av Swedish Algae Factory, ett företag som odlar kiselalger och producerar Algica – ett material som används i hudvårdsprodukter.

Faktorer såsom pH, temperatur, konduktivitet, och solljus analyserades med både linjär regression och XGBoost för att identifiera vilka faktorer som hade störst påverkan på skördens storlek och renhet. Resultaten visar att inga starka faktorer är möjliga att hitta för att förstå tillväxten av kiselalger. Däremot kunde man observera att kontaminationen av grönalger ökar med temperaturen, och att den samlade renheten minskar över tid.

Eftersom datamängden var relativt liten är osäkerheterna stora. Dessutom bidrar korrelationer mellan parametrarna till att datasetet uppvisar mindre variation. Det finns också många variationer som inte kan åskådliggöras med den nuvarande insamlade data. Därav föreslår studien en förändrad metod för datainsamling där både skörd och parametrar delas upp i moduler för att både öka mängden data och mäta variationer i systemet. Detta kan leda till mer komplexa tidsberoende analyser av tillväxt och öka tillförlitligheten.

Abstract

Machine learning has emerged as a key technology with wide-ranging applications, particularly as access to large datasets and computational power has rapidly increased. This development has enabled the use of more powerful models to analyze complex systems and make predictions.

The aim of this study is to investigate biotic and abiotic factors affecting the cultivation of diatoms using various machine learning methods, with a specific focus on understanding growth dynamics and reducing the contamination of green algae in the cultivation system. The data was collected by Swedish Algae Factory, a company that cultivates diatoms and produces Algica – a material used in skincare products.

Factors such as pH, temperature, conductivity, and sunlight were analyzed using both linear regression and the machine learning model XGBoost, to identify which variables had the greatest impact on yield and purity. The results indicate that no strong predictors could be identified to explain the growth of diatoms. However, it was observed that green algae contamination increases with temperature, and that the overall purity in the system declines over time as a result of increasing contamination.

Due to the relatively small dataset, the uncertainties are significant, and correlations between parameters contribute to a limited variability in the data. Moreover, many system variations could not be captured with the available data. Therefore, the study proposes a revised method for data collection, where both yield and environmental parameters are recorded separately for each module. This would not only increase the data volume but also make it possible to detect variations within the system. Such an approach may enable more complex time-dependent analyses of growth and improve the reliability of the results.

Innehåll

1 Inledning

Termen maskininlärning myntades 1959 av Arthur Samuel för att beskriva datorers förmåga att lära sig mönster från data utan att vara explicit programmerade för detta [2]. Det som då var av begränsat akademiskt intresse har i dag blivit en av de mest centrala teknikerna inom en rad branscher, från finans och sjukvård till miljöteknik. Den snabba utvecklingen av datorteknik och tillgängligheten av stora, komplexa dataset har ytterligare påskyndat användningen av maskininlärning som metod för att förstå och förutsäga skeenden i allt mer dynamiska system.

I denna studie undersöks hur maskininlärning kan tillämpas för att analysera olika biotiska och abiotiska faktorer som påverkar algtillväxt. Projektet är särskilt relevant för företaget Swedish Algae Factory, som samlar in data om algtillväxt under varierande förhållanden och eftersträvar en mer effektiv och hållbar odlingsprocess. Från algerna utvinns kiseldioxid som kan användas inom flera områden, som för att öka effektiviteten av solpaneler samt inom hudvård. Eventuellt kan resultatet också appliceras på odling av andra organismer.

1.1 Bakgrund

Swedish Algae Factory är ett svenskt företag som grundades 2016 [3]. Idag producerar de Algica, ett material som utvinns ur skalen på kiselalger. Algica används inom hudvård, som en fuktighetsbevarande och rengörande ingrediens, men kan även utnyttjas som SPF-booster [4]. Dessutom kan kiselalger användas för att öka effektiviteten hos solpaneler [5].

Kiselalgerna odlas i moduler, och målet är att cirkulärt använda vatten från en lokal fiskodling som näring för algerna. Efter att algerna har renat vattnet skickas det numera syrerika vattnet tillbaka till fiskodlingen, och kan sedan återigen användas som näring för algerna. Biomassan som blir kvar efter att Algica har utvunnits används för att antingen värma upp anläggningen eller för att producera gödsel [6]. Dessutom binds minst 8 kg koldioxid, 1 kg kväve och 0,1 kg fosfor in vid produktionen av 1 kg Algica.

Utöver de tidigare nämnda användningsområdena kan man också använda kiselalger för att göra biodiesel, men med nuvarande teknik är det alldeles för dyrt [7]. Man kan också använda kiselalger för att ersätta majs och soja som djurfoder [8]. Ytterligare en intressant tillämpning är inom läkemedel, eftersom kiselalger har bevisad effekt mot inflammation och cancer [9].

1.2 Syfte

Syftet med denna studie är att använda olika maskininlärningsmodeller för att identifiera och förutsäga de biotiska och abiotiska faktorer som främst påverkar kiselalger och grönalgers tillväxt. Arbetet bygger på data från Swedish Algae Factory och ska dessutom resultera i förslag på möjliga förbättringar av den nuvarande odlingsprocessen.

2 Teori

Nedan följer en kort introduktion till maskininlärning, och därefter en beskrivning av de olika maskininlärningsmodeller som har använts. Därpå följer en förklaring av en metod som förklarar hur maskininlärningsmodellen gör sin prediktion. Sist kommer en kort text om kiselalger.

2.1 Introduktion till maskininlärning

Man kan dela in maskininlärning i tre huvudsakliga områden: supervised learning, reinforcement learning och unsupervised learning [10]. I detta arbete används enbart supervised learning, där datorsystemet får dataexempel med etiketter och uppgiften är att skapa en funktion som kopplar indatan till utdatan. I reinforcement learning får systemet istället återkoppling på sina handlingar, och i unsupervised learning får systemet varken återkoppling eller data med etiketter.

Supervised learning kan i sin tur delas in i regressions- och klassificeringsalgoritmer [10]. Regressionsalgoritmer har som uppgift att försöka uppskatta sambandet mellan indatavariabler och kontinuerliga eller numeriska utdatavariabler. Klassificeringsalgoritmer syftar till att dela in datan i olika kategorier, och används till exempel för kategorisering av bilder. I detta arbete kommer inte klassificeringsalgoritmer användas, utan istället kommer bland annat linjär regression användas eftersom att det passar den tillgängliga data bättre.

När man använder maskininlärning brukar man dela upp data i träningsdata och testdata för att validera modellen. Inom maskininlärning är en modell en matematisk representation av en riktig process [10]. Normalt sett plockas ungefär 30 % slumpmässigt ut för att användas som testdata, och resterande 70 % är träningsdata. Man tränar sin modell på träningsdatan, och testar sedan hur väl den fungerar på testdatan.

Två problem man vill undvika med modellerna är överanpassning (eng. overfitting) och underanpassning (eng. underfitting). Överanpassning är när modellen passar träningsdata väldigt bra, men inte kan generalisera till testdata [10]. Underanpassning är istället när modellen inte alls passar träningsdata, vilket kan bero på att den är för enkel och inte kan beskriva sambandet mellan indata och utdata.

2.2 Linjär regression

Linjär regression är en enkel typ av regressionsalgoritm. Det är en parametrisk modell, vilket innebär att modellen antar en ändlig uppsättning parametrar [10].

Vanlig linjär regression undersöker enbart sambandet mellan en indatavariabel och utdatavariabeln, enligt ekvation 1, där y är utdatavariabeln, x är indatavariabeln och β_0 samt β_1 är koefficienter [11].

$$y = \beta_0 + \beta_1 x \tag{1}$$

Den här ekvationen förutsätter dock att sambandet är linjärt, vilket inte nödvändigtvis stämmer. Eftersom den också bara innehåller en indatavariabel kan modellen vara för enkel och andra faktorer som påverkar resultatet saknas [11]. Därför kan man istället använda multipel linjär regression, där flera indatavariabler ingår. Då använder man istället ekvation 2, där q är antalet indatavariabler.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_q x_q \tag{2}$$

Om vi har n exempel kan vi istället skriva ekvationen på matrisform:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} \ldots\ldots x_{1q} \\ 1 & x_{21} & x_{22} \ldots\ldots x_{2q} \\ \vdots & \vdots & \vdots \quad\quad \vdots \\ 1 & x_{n1} & x_{n2} \ldots\ldots x_{nq} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_q \end{bmatrix} \iff y = X\beta \tag{3}$$

För att optimera värdena på β kan man använda minstakvadratmetoden, som syftar till att minimera felkvadratsumman, det vill säga summan av de kvadrerade felen (eng. SSE: sum of squared errors) [11]. Felet är skillnaden mellan det uppskattade värdet $\hat{y}$ och det verkliga värdet y. Hur felkvadratsumman beräknas syns i ekvation 4, där $\hat{y_i} = X_i\hat{\beta}$ och $X_i = [1\ x_{i1}\ x_{i2}\ \cdots\ x_{iq}]$.

$$SSE = \sum_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{4}$$

De optimerade värdena på β tas fram genom pseudoinversen för matrisen, som används för icke-kvadratiska matriser. Då får man $\beta_{opt} = (X^T X)^{-1}X^T y$ [11].

Sedan måste också uppskattningen utvärderas. Det kan göras med SSE, men man kan också beräkna summan av absolutfelen (eng. SAE: sum of absolute errors) [11]. Det beräknas enligt ekvation 5:

$$SAE = \sum_{i=1}^{n}|y_i - \hat{y_i}| \tag{5}$$

Slutligen kan man också beräkna R^2-värdet med hjälp av regressionskvadratsumman (eng. SSR: square sum of regression) [11].

$$R^2 = \frac{SSR}{SSE + SSR},\ SSR = \sum_{i=1}^{n}(\hat{y_i} - \overline{y})^2 \tag{6}$$

$\overline{y}$ är medelvärdet som beräknas enligt $\overline{y} = \frac{\sum_{i=1}^{n} y_i}{n}$.

För att jämföra olika modeller kan man också jämföra deras kvadratiska medelvärden (eng. RMS: root mean square). Det beräknas genom att ta roten ur medelvärdet av felkvadratsumman, enligt ekvationen nedan [12].

$$RMS = \sqrt{\frac{SSE}{n}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2} \tag{7}$$

2.3 XGBoost (Extreme Gradient Boosting)

XGBoost är en högeffektiv implementation av gradientförstärkta beslutsträd som är optimerad för både hastighet och minnesanvändning [10]. Den hanterar saknade värden automatiskt, kan arbeta direkt med glesa matriser och har inbyggd både ℓ_1- och ℓ_2-regularisering, vilket minskar risken för överanpassning. Biblioteket integreras sömlöst med `scikit-learn` och stödjer parallell träning, kolumnsubsampling, inlärningstakt och tidig stoppning [13].

Ett beslutsträd är en typ av maskinlärningsmodell som består av flera delar [10]. Den första är en startnod, eller rot, som inte har några inkommande grenar. Det finns också interna noder, som är märkta med namnet av en attribut (eng. feature) och har en inkommande gren samt två eller fler utgående grenar. I slutet av varje väg finns bladnoder, också kallade beslutsnoder, som är märkta med en klass. Grenarna utgår från de interna noderna, och varje gren är märkt med ett värde eller intervall som är möjligt för nodens attribut att anta.

Algoritmen bygger på att successivt lägga till många små, grunda träd där varje nytt träd försöker korrigera de fel som den kumulativa modellen fortfarande gör [14]. Nedan följer en översiktlig skiss av algoritmen:

Låt $F_0(x) = \arg\min_\theta \sum_i L\big(y_i, \theta\big)$ vara en konstant basmodell
för $m = 1, \ldots, M$
  beräkna pseudoresidualerna r_i för samtliga observationer
  träna ett grunt träd f_m på residualerna

uppdatera $F_m = F_{m-1} + \nu f_m$
returnera F_M

Pseudoresidualen definieras som den negativa gradienten av förlustfunktionen med avseende på den aktuella modellen

$$r_i \;=\; -\left.\frac{\partial L\big(y_i, F(x_i)\big)}{\partial F(x_i)}\right|_{F=F_{m-1}} . \tag{8}$$

Till skillnad från traditionell gradient boosting använder XGBoost en andragradig Taylor-approximation av objektivfunktionen

$$L\big(y, F_{m-1} + \eta f_m\big) \simeq L\big(y, F_{m-1}\big) + g_i\,\eta f_m(x_i) + \tfrac{1}{2} h_i\,(\eta f_m(x_i))^2, \tag{9}$$

med $g_i = \partial_{F_{m-1}} L\big(y_i, F_{m-1}(x_i)\big)$ och $h_i = \partial^2_{F_{m-1}} L\big(y_i, F_{m-1}(x_i)\big)$. Den andra ordningens information gör att modellen ofta konvergerar med färre iterationer.

Träden f_m hålls avsiktligt grunda (t.ex. högst 6 nivåer) för att begränsa modellens kapacitet och därigenom minska överanpassning. Vid konstruktionen kan man justera bland annat trädets djup, hur stor andel av träden som får se varje observation (eng. row subsampling) respektive varje variabel (eng. column subsampling). Bladvikterna bestäms analytiskt som

$$w_j \;=\; -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda}, \tag{10}$$

vilket minimerar den regulariserade objektivfunktionen

$$\mathrm{Obj}(F) = \sum_i L\big(y_i, \hat{y}_i\big) + \sum_{k=1}^{M} \Big(\gamma T_k + \tfrac{1}{2}\lambda \|w^{(k)}\|_2^2 + \alpha \|w^{(k)}\|_1\Big) \tag{11}$$

där T_k är antalet blad i träd k, λ styr ℓ_2-straffet på bladvikterna, α styr ℓ_1-straffet på bladvikterna och γ är kostnaden för varje nytt blad.

Antalet träd M och inlärningstakten ν fungerar som kraftfulla regulariseringsparametrar: fler träd och större ν ökar modellens komplexitet, medan tidig stoppning (eng. early stopping) på en valideringsmängd kan användas för att automatiskt avbryta träningen innan överanpassning uppstår.

När man bygger en maskininlärningsmodell är det också viktigt att anpassa hyperparametrarna. Detta gäller särskilt för trädmodeller, eftersom de har många hyperparametrar [15]. En hyperparameter är en parameter som inte kan uppskattas direkt från datan. Istället måste hyperparametrarna bestämmas innan träningsprocessen av modellen eftersom de bestämmer själva uppbyggnaden av modellen. På grund av detta är det viktigt att optimera hyperparametrarna för att få en så bra modell som möjligt.

En metod som används för hyperparameteroptimering är slumpmässig sökning (eng. random search) [15]. Slumpmässig sökning innebär att ett förutbestämt antal prov slumpmässigt väljs ut inom ett visst intervall. Sedan undersöks alla proven tills försökstiden tar slut, och de bästa proven antas som hyperparametrar. Med ett tillräckligt stort antal prov borde detta resultera i ett väldigt bra hyperparametervärde. Denna metod har visat sig vara bättre än den klassiska metoden rutnätssökning (eng. grid search) [16].

2.4 SHapley Additive exPlanations, SHAP

Inom maskininlärning är det viktigt att förstå hur eller varför maskininlärningsmodeller kommer fram till sin prediktion. SHAP är en metod som tilldelar varje attribut ett bidragande värde för varje exempel i träningsdata [17]. SHAP-värdet till attribut i för ett exempel $x = (x_1, \ldots, x_n)$ väljs som det förändrade väntevärdet för modellen, $f(x)$, ger prediktionen betingat på x_i. Alltså är SHAP-värdet $\phi_i = E[f(z)|z_i = x_i]$. SHAP-värden beskriver hur man går från förväntade prediktionsvärdet, $E[f(z)]$ till modellens prediktion $f(x)$.

För att beräkna SHAP-värden håller man ett värde statiskt och sedan väljer alla kombinationer i träningsdata på andra attribut [17]. Detta är något som kan vara beräkningstungt för mycket data och stora modeller och därav har olika modelspecifika approximationer tagits fram. Fördelar med SHAP är att det fungerar på alla modeller och har förmåga att ge insikt på lokal nivå för enstaka datapunkter men också få inblick i modellen för hela träningsdata.

2.5 Kiselalger

Kiselalger, även kallade diatoméer, står för ungefär 20 procent av fotosyntesen på jorden [18]. Namnet diatom kommer från det grekiska diatomos, som betyder delad på mitten. Detta syftar till att kiselalger har två separata cellväggar uppbyggda av kiseldioxid. Det finns cirka 200 000 olika arter, och diversiteten mellan arter är stor. Storleken skiljer sig från några mikrometer till en millimeter. Dessutom finns det arter som består av kedjor av celler istället för enskilda celler.

Det som gör kiselalger speciella är just deras cellvägg av kiseldioxid, den så kallade frustulen. Mellan arter så skiljer sig både formen och mönstret av porerna i frustulen, vilket gör att man ofta får vackra mikroskopbilder där man kan se den stora diversiteten mellan de olika arterna (se Figur 1). Traditionellt har man delat in kiselalgerna i två grupper utifrån frustulens symmetri [19]. Centriska kiselalger är runda med radiell symmetri, och pennata kiselalger är mer avlånga och har bilateral symmetri.

Strukturen av frustulen påverkar upptagningen av både näring och ljus [19]. Molekylerna och generna som bygger upp frustulen påverkas i sin tur av olika begränsningar på miljön, som ljusintensitet och tillgång på näringsämnen. Det har visats att tillgängligheten och koncentrationen av kisel och andra näringsämnen kan begränsa tillväxten och celldelningen av kiselalger, samt påverka fysiologin och de mekaniska egenskaperna av frustulen. En ansamling av lipider i frustulen har observerats på grund av brist på kväve och fosfor, vilket kan påverka konstruktionen av cellväggen. Dessutom har man observerat en ökad mängd kisel i cellen som ett resultat av en begränsad mängd av järn, kväve, samt fosfor.

3 Metod

I detta kapitel presenteras den metodik som använts för att bearbeta och analysera data relaterad till algodling. Arbetet omfattar den förbehandling som krävs av data samt tillämpning med hjälp av statistiska och maskininlärningsmetoder för att förstå algodlingens växtfaktorer.

3.1 Datainsamling

Data som används för denna studie har erhållits från Swedish Algae Factory som bedriver algodling i kontrollerade inomhusmiljöer. Mätvärden samlas in med automatiska sensorer och manuella värden i veckorapporter. Datainsamlingen är främst uppdelad i tre huvudsakliga dataset:

1. **Odlingsmoduler (OM-A och OM-B)**

   Mäter odlingsparametrar i en systemtank vars vatten sedan pumpas runt till odlingsbanorna.

2. **Systemfil**

   Innehåller ytterligare parametrar från odlingarna som solljus och konsumerad kWh för hela systemet.

3. **Odlingsresultat**

   Dagvis sammanställning av parametrar från odlingsmodulerna och systemfilen samt veckovis rapportering av olika skördevärden.

Data för moduler och skördar sträcker sig över en period på cirka 12 månader under delar av 2023 och 2024, medan systemfilen har data för cirka 5 månader. Formatet på data i odlingsmoduler och systemfilen är CSV. Odlingsresultat har formatet xlsx men konverterades till CSV.

3.2 Databeskrivning

Datafilernas parametrar är olika och varierar i tillförlitlighet samt användningsbarhet. Nedan följer en beskrivning av de olika dataseten och parametrarna dessa innehåller.

3.2.1 Odlingsmoduler (OM-A och OM-B)

Varje odlingsmodul innehåller 38 odlingsbanor och mätvärdena kommer från en systemtank. Värdena på parametrarna mäts med olika frekvenser.

- **pH-värde:** Mäts i systemtanken vars innehåll sedan pumpas ut till odlingsytor. Värdet påverkas av koldioxidtillförsel och algernas fotosyntetiska aktivitet. Under ljusexponering konsumerar algerna koldioxid vilket höjer pH-värdet, medan respiration sker i mörker och sänker pH-värdet. Systemet reglerar pH-värdet genom koldioxidtillförsel för att upprätthålla eftertraktade odlingsförhållanden.
- **Tillförd CO_2:** Ett beräknat värde för koldioxidtillförsel vid tre doseringspunkter i respektive modul. Värdet bedöms ha viss osäkerhet på grund av manuella justeringar som inte alltid dokumenterats.
- **Konduktivitet:** Mått på systemtankens salthalt. Systemet eftersträvar ett konstant värde genom automatisk påfyllning av färskvatten för att kompensera för avdunstning.
- **Temperatur:** Mäts i systemtankens botten där temperaturen är mest stabil. Noterbart är att temperaturskillnader på upp till ett par grader kan förekomma mellan olika delar av systemet, där de översta odlingsbanorna uppvisar störst dygnsvariation.

Mätningsfrekvenserna på parametrarna är olika. Det finns också perioder i data som helt eller delvis saknar värden. Ibland är dessa avbrott någon timme men kan sträcka sig upp emot en vecka. I Figur 2 i appendix A.1 syns hur parametrarna varierar över tid.

3.2.2 Systemfil

Denna fil innehåller aggregerade och beräknade värden för hela systemet. I Figur 3 i appendix A.1 syns ett exempel på hur parametrarna varierar över tid.

- **Total CO_2-förbrukning:** Den ackumulerade koldioxidförbrukningen över tre doseringspunkter.
- **Medeltemperatur 24h:** Ett glidande medelvärde över 24 timmar baserat på flera temperaturgivare i systemet.
- **Tempsumma:** Ackumulerad temperatur under dygnet.
- **LUX 1h:** Ljusmätare placerad i växthusets tak. Automatiska gardinsystem används för att begränsa ljusexponering och förhindra algstress.
- **kWh per dag:** Energiförbrukning för odlingsbelysningen.

3.2.3 Odlingsresultat

Detta dataset innehåller två olika delar. Sammanställda värden dagvis om hela systemet samt veckovisa sammanställningar om skördad mängd:
Den dagliga sammanställningen har värden för:

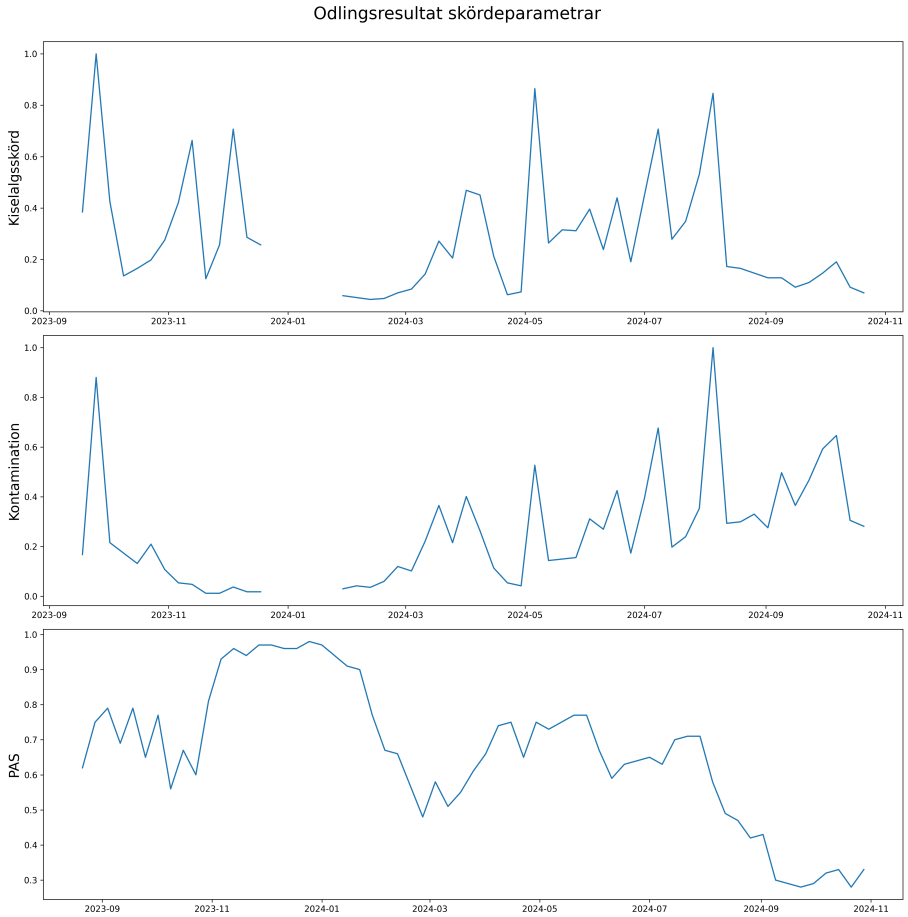
- **Ljus %:** Beräknad inställning av odlingsbelysning, procent av maximal effekt.
- **Ljus i watt:** Ljus % omräknat till watt.
- **Beräknad korrigerad kWh per dygn:** Beräknad elförbrukning per dygn.
- **Mätt ljuseffekt kWh per dag:** Uppmätt elförbrukning per dygn.
- **tot Ljus kWh:** Beräknad mängd ljus per dygn, med hänseende till odlingsbelysning samt solljus.
- **Ljussumma Sol LUX:** Ackumulerat solljus som når växthuset under ett dygn.
- **temp Vmedel:** Medeltemperatur över dygnet.
- **temp Vmax:** Maximal uppmätt temperatur under dygnet.
- **temp Vmin:** Minimal uppmätt temperatur under dygnet.
- **Temp Summa:** Ackumulerad temperatur under varande dygn. Är samma som medeltemperaturen x 24.

Skördeparametrarna som registreras manuellt:

- **Kiselalgsskörd:** Den totala volymen skördade kiselalger.
- **Kontamination:** Volym av oönskade mikroorganismer i skörden, ofta grönalger.
- **Procent användbar skörd (PAS):** Andelen av den totala skörden som består av kiselalger, det vill säga den delen som inte består av kontamination.

Skörd sker varje vecka men endast en tredjedel av systemet skördas per vecka. Alltså går det i 3-veckors scheman, men just vilken tredjedel av odlingsbanorna som skördas varje vecka mäts inte.

Data för odlingsresultat och odlingsmoduler finns för 55 veckor, systemfilen för 23 veckor. Hur de olika skördeparametrarna kiselalgsskörd, kontamination och PAS varierar över de 55 veckor som data finns syns i Figur 2. Hur de andra parametrarna varierar syns i Figur 1 i appendix A.1. Både kiselalgsskörd och kontamination saknar värden under slutet av 2023 och början av 2024, medan det finns värden för PAS under hela perioden.



Figur 2: Variation över tid i skördeparametrarna kiselalgsskörd, kontamination och PAS, baserat på insamlad data. Parametrarna är normerade så att varje värde är delat med det maximala uppmätta värdet.

3.3 Databehandling

Data med för hög upplösning snittades över dagar och sammanställdes till treveckors genomsnitt för parametrarna. Datumkolumnen ersattes med en kolumn som innehöll tid från första datapunkt. Sedan skapades två träningsdataset, ett för x-data och ett för y-data. Y-datasetet innehöll kiselalgsskörd, kontamination och PAS för varje inkluderad skörd, och x-datasetet innehöll alla andra inkluderade parametrar för de respektive skörderna, med matchande index. I detta steg av databehandlingen exkluderades alla punkter där något parametervärde saknades. Parametrarna som inkluderades i x-data varierade beroende på försök.

Slutligen normerades data genom division med det högsta värdet kolumnvis. Detta gjordes framför allt för att enklare kunna jämföra hur högt modellen värdesatte olika variabler.

3.4 Statistisk analys och linjär regression

För att undersöka samband mellan olika skördeparametrar användes korrelationsanalys. Specifikt analyserades sambandet mellan valda parametrar och skörd av kiselalger, kontaminering samt procent användbar skörd (PAS). Pearsons korrelationskoefficient användes för att kvantifiera styrkan och riktningen av linjära samband mellan variablerna. Därefter tränades linjära regressions modeller för samma parametrar. Alla beräkningar utfördes i Python version 3.11.9.

3.5 Modellträning och analys med XGBoost

XGBoost-modeller tränades på olika konfigurationer av data, där varje konfiguration innehöll minst fem x-variabler och en av tre olika y-variabler. Denna y-variabel var antingen kiselalgsskörd, kontamination eller PAS.

Hyperparametrarna för att träna modellerna valdes antingen som modellens standardvärden eller genom slumpmässig hyperparameteroptimering. För varje steg i den slumpmässiga optimeringen testades värden utifrån normalfördelningar vars medelvärden var standardvärdet för hyperparametern och standardavvikelser på mellan en tiondel och en tredjedel av medelvärdet beroende på parameter.

En del av datasetet sattes som valideringsset, vilket modellen aldrig skulle få stöta på före sin slutliga validering. Resterande data användes för att träna modeller med olika hyperparametervärden, valda slumpmässigt från de tidigare nämnda normalfördelningarna. I varje runda sorterades modellerna efter median av RMS-fel, och den som presterade bäst blev grunden för nästa runda genom att dess värden sattes som medelvärden för hyperparametrar i nästa omgång. Detta upprepades ett antal gånger med minskande standardavvikelser för varje runda och till slut tränades hyperparameterkombinationen som presterat bäst dittills på all data som inte ingick i valideringssetet för att sedan testas på detta.

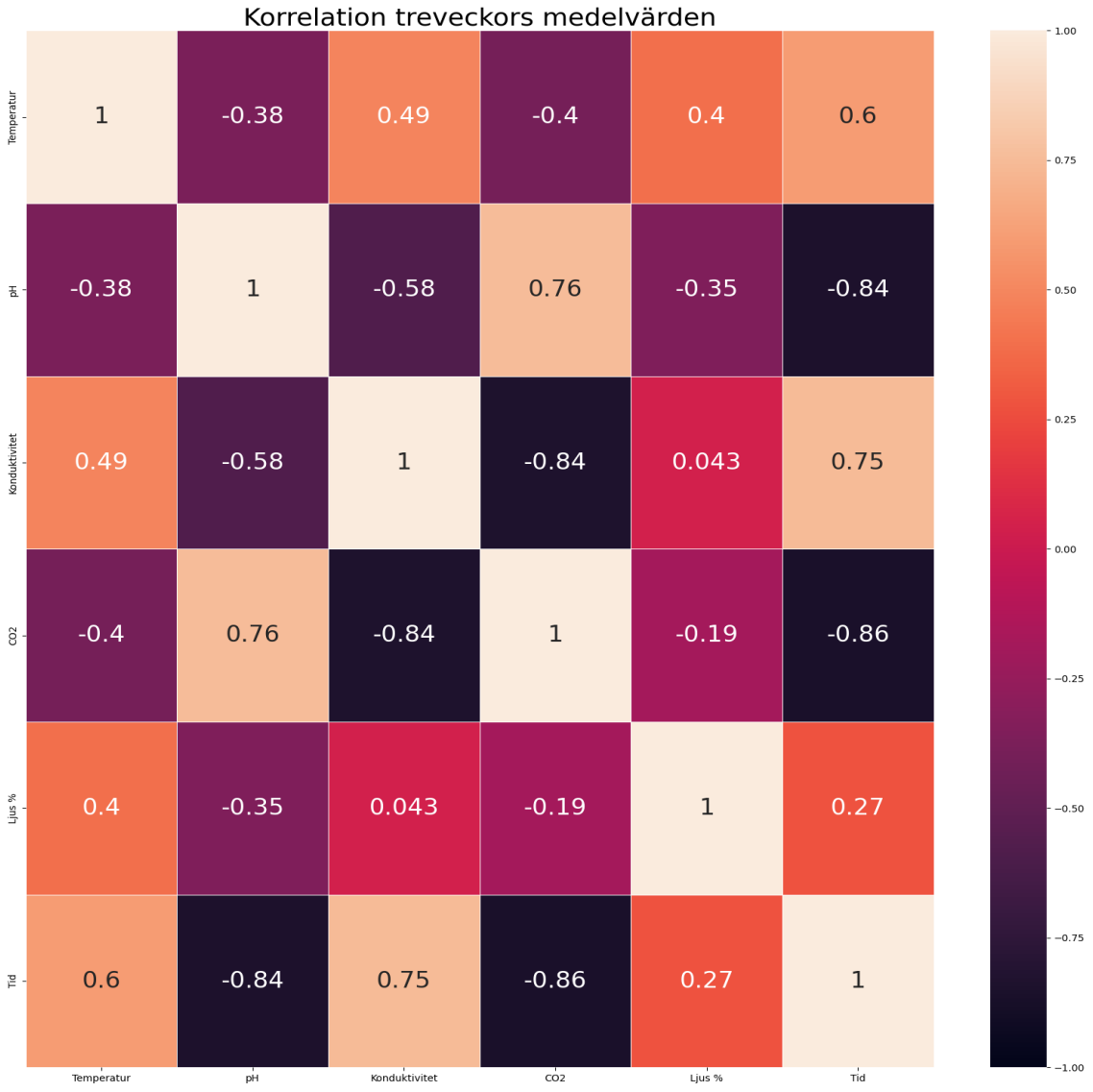
Därefter analyserades parametrarnas relevans för de tränade XGBoost-modellerna med SHAP. För att visualisera resultatet genererades två olika typer av figurer. Den ena användes för att rangordna genomsnittliga absoluta SHAP-värden och den andra för att visa hur varje observation påverkar SHAP-värdet.

4 Resultat

I detta avsnitt presenteras först resultaten från den inledande databehandlingen. Därefter redovisas resultaten från de tillämpade linjära analyserna och XGBoost. Analysen fokuserar på att markera de starkaste sambanden som observeras mellan skörd och parametrar.

4.1 Databehandling

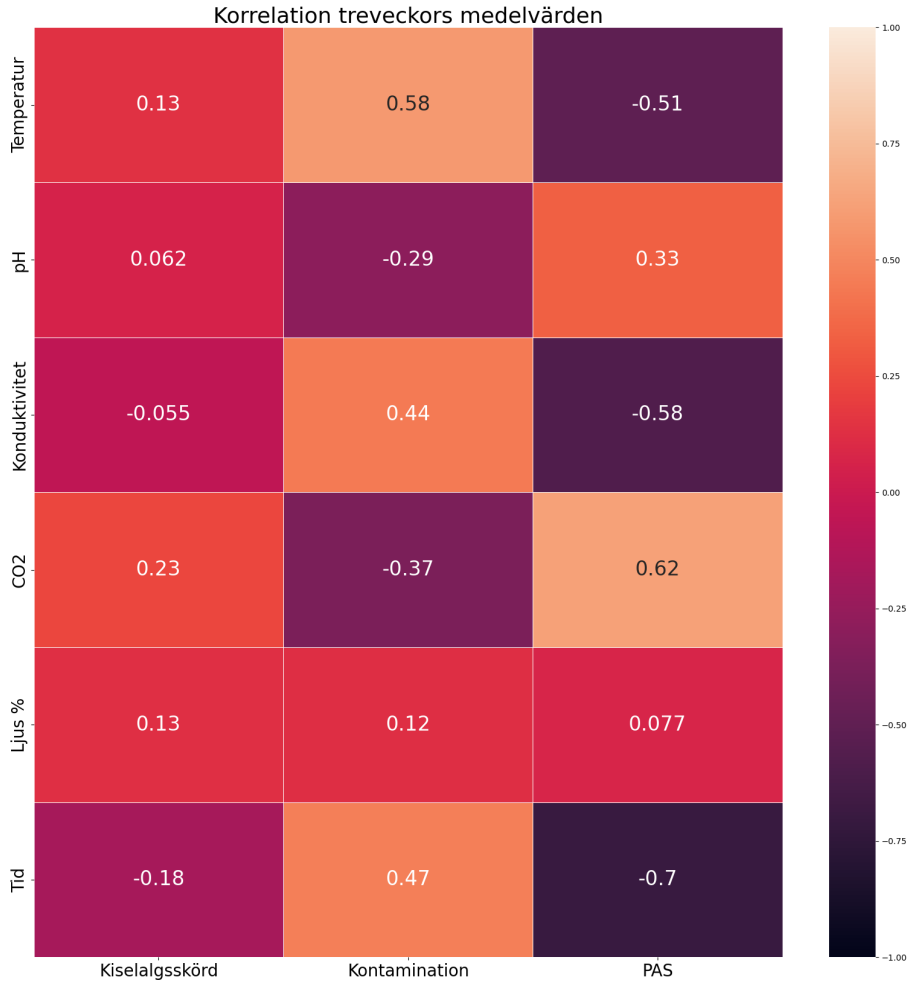
Den bearbetade treveckors sammanställda data visar trender i hur algodlingens parametrar för temperatur, pH, CO_2 och konduktivitet varierar över tid enligt Figur 4 i Appendix A.1. Korrelationsanalys mellan parametrarna visar att det finns starka korrelationer mellan vissa parametrar (Figur 3). Den parameter som har flest starka korrelationer är tid, som korrelerar mot temperatur, pH, konduktivitet och CO_2 med styrkor $0,6$, $-0,84$, $0,75$ och $-0,86$. Den parameter som korrelerar minst med andra är ljus % som uppvisar svagare trender, med störst värde på 0,4 mot temperatur.



Figur 3: Korrelation mellan valda parametrar temperatur, pH, konduktivitet och CO_2 från odlingsmodul samt ljus % från odlingsresultat. Data för beräkning av korrelationen är tagen som treveckors medelvärde bakåt i tiden. Tid-parametern är satt som datumet på den måndag efter de tre veckorna, det vill säga dagen för skörd.

4.2 Korrelation och linjära samband

Utefter den data som togs fram gavs korrelationen för att finna linjära samband mellan parametrarna (Figur 4). Där syns att vissa parametrar inte visade på någon korrelation med de olika skördeparametrarna, som exempelvis pH och kiselalgsskörd. Andra parametrar hade däremot starkare korrelationer, som temperatur och kontamination. Linjära regressionsmodeller mellan valda parametrar och skördeparametrarna gav liknande resultat som korrelationsundersökningen (se appendix A.2 Figur 5, 6 & 7).



Figur 4: Pearson-korrelationen mellan skördeparametrarna kiselalgsskörd, kontamination och PAS mot parametrarna temperatur, pH, konduktivitet, CO_2, ljus % och tid. En mörk färg innebär en starkt negativ korrelation, medan en ljus färg innebär en starkt positiv korrelation.

4.2.1 Svaga linjära samband mellan undersökta parametrar och kiselalgsskörd

I korrelationsanalysen för kiselalgsskörd kan inte några starka linjära samband hittas (Figur 4). Alla värden på korrelationer är under 0.25 och visar att det finns obetydande korrelationer mellan de olika undersökta parametrarna och kiselalgsskörd.

4.2.2 Kontamination korrelerar starkast med temperatur

Kontaminationen visar flera linjära samband som uppvisas med måttlig styrka (Figur 4). Starkast korrelation finns för temperatur med koefficient 0,58. Tid och CO_2 visar också på måttliga korrelationer, med värden på 0,47 respektive -0,37. Alla parametrar utom ljus % uppvisar svaga linjära trender. Ljus % har en koefficient på 0,12 och visar inte på någon linjär trend.

4.2.3 PAS uppvisar flera starka korrelationer till de undersökta parametrarna

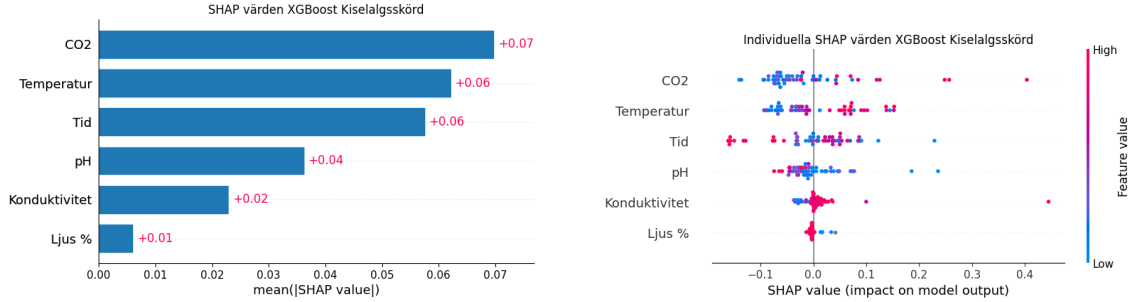
Procent användbar skörd (PAS) är den skördeparameter som uppvisar starkast korrelationer (Figur 4). Den starkaste korrelationen finns till tid, där värdet är -0,7. Även CO_2, konduktivitet och temperatur visar på relativt starka linjära samband. Däremot har ljus % en väldigt liten koefficient på 0,077, så det finns inget linjärt samband mellan PAS och ljus %.

4.3 Analys med XGBoost

Resultaten från de utvecklade XGBoost-modellerna presenteras nedan. Resultatet är uppdelat efter skördeparameterarna kiselalgsskörd, kontamination och PAS.

4.3.1 Flera parametrar var viktiga för prediktionen av kiselalgsskörd

Påverkan av olika variabler på XGBoost-modellens prediktion av kiselalgsskörd visas i analyserna (Figur 5a & 5b). Variablerna rangordnas efter sina genomsnittliga absoluta SHAP-värden, där CO_2 uppvisar störst inverkan medan ljus % har minst (Figur 5a). Höga nivåer av CO_2, konduktivitet och temperatur är generellt associerade med positiva SHAP-värden och därmed en ökad förutsagd skörd, medan lägre nivåer bidrar negativt till modellens prediktion (Figur 5b). För CO_2 är skillnaden mellan höga och låga nivåer särskilt uttalad, medan konduktivitet och temperatur uppvisar en mer symmetrisk påverkan. Tiden tenderar däremot att verka i motsatt riktning: längre tider ger negativa och kortare tider positiva SHAP-värden. Effekten av pH är asymmetrisk - låga pH-värden ökar SHAP-värden mer än vad höga pH-värden minskar den, d.v.s. modellen är känsligare för sura förhållanden än basiska. Slutligen är ljusintensitetens inverkan genomgående liten; dess SHAP-värden är tätt centrerade kring noll och bidrar därmed minimalt till modellens utslag.

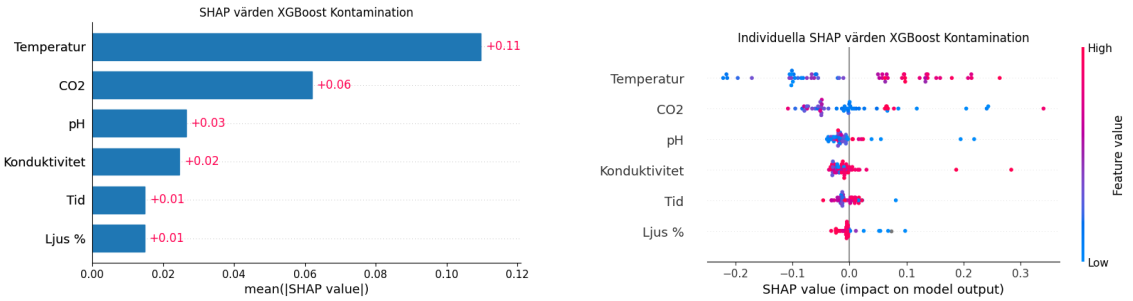


(a) (b)

Figur 5: SHAP-analyser av XGBoost-modellen för förutsägelse av kiselalgsskörden. Panel (a) är genomsnittliga absoluta SHAP-värden som visar varje variabels genomsnittliga betydelse för modellens prediktion. Panel (b) är individuella SHAP-värden som visar hur varje observation (punkt) bidrar till modellens utdata. Färgskalan visar om värdet på variabeln är högt (rött) eller lågt (blått), vilket ger insikt i hur höga eller låga värden påverkar sannolikheten för kontaminering.

4.3.2 Temperatur var avgörande för kontaminationsprediktion

På liknande sätt som i föregående avsnitt illustreras variablernas påverkan på XGBoost-modellens prediktion av kontamination i de följande analyserna (Figur 6a & 6b). De genomsnittliga absoluta SHAP-värdena visar att temperatur har störst inverkan på modellens prediktion, följt av CO_2, pH och konduktivitet (Figur 6a). Tid och ljus % har däremot minst betydelse. Riktningen på effekterna illustrerar att höga värden på temperatur och konduktivitet tenderar att ge positiva SHAP-värden, vilket innebär att de bidrar till en högre förutsagd kontaminationsnivå – och denna effekt är starkare för höga värden än motsvarande negativa påverkan från låga värden (Figur 6b). Ljus % uppvisar en motsatt och asymmetrisk effekt, där låga värden (mindre ljus) påverkar modellen mer positivt än vad höga värden (mer ljus) påverkar negativt. Övriga variabler uppvisar stor osäkerhet i sina SHAP-värden, vilket gör det svårt att dra säkra slutsatser om deras påverkan.

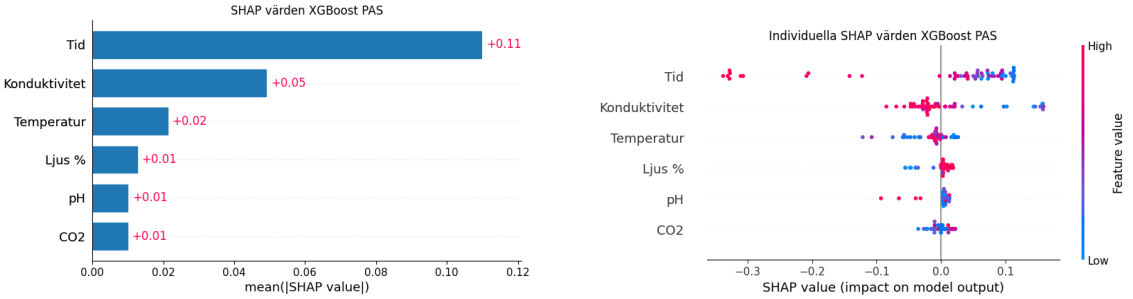


(a) (b)

Figur 6: SHAP-analyser av XGBoost-modellen för förutsägelse av kontaminering. Panel (a) är genomsnittliga absoluta SHAP-värden som visar varje variabels genomsnittliga betydelse för modellens prediktion. Panel (b) är individuella SHAP-värden som visar hur varje observation (punkt) bidrar till modellens utdata. Färgskalan visar om värdet på variabeln är högt (rött) eller lågt (blått), vilket ger insikt i hur höga eller låga värden påverkar sannolikheten för kontaminering.

4.3.3 Tid var den främsta variabeln för att prediktera PAS

Effekten av PAS på XGBoost-modellen framgår av de presenterade analyserna (Figur 7a & 7b). Rangordningen av variablerna visar att tid är den mest inflytelserika faktorn, följd av konduktivitet och temperatur, medan ljus %, CO_2 och pH har mindre betydelse (Figur 7a). Riktningen på effekterna visar att höga värden för tid ger starkt negativa SHAP-bidrag, vilket indikerar en koppling till lägre förutsagda PAS-nivåer, medan låga tidsvärden har en svagt positiv inverkan (Figur 7b). För pH gäller en liknande asymmetri – högt pH drar ned modellens prediktion mer än vad lågt pH höjer den. Konduktivitet och temperatur uppvisar också en övervikt av negativa effekter, dock med hög osäkerhet för temperatur. De återstående variablerna ljus % och CO_2 tenderar däremot att ha positiva effekter, vilket syns i att högre värden på dessa ofta bidrar till en ökad prediktion med en symmetrisk spridning av SHAP-värden kring noll.



(a) (b)

Figur 7: SHAP-analyser av XGBoost-modellen för förutsägelse av PAS. Panel (a) är genomsnittliga absoluta SHAP-värden som visar varje variabels genomsnittliga betydelse för modellens prediktion. Panel (b) är individuella SHAP-värden som visar hur varje observation (punkt) bidrar till modellens utdata. Färgskalan visar om värdet på variabeln är högt (rött) eller lågt (blått), vilket ger insikt i hur höga eller låga värden påverkar sannolikheten för kontaminering.

4.3.4 Modeller presterade bäst när de förutspådde PAS

XGBoost-modellernas förutspående förmåga varierade beroende på vilken skördparameter som skulle förutspås. Lägst RMS-fel, på normerade dataset, kunde uppnås då den beroende variabeln var PAS och högst då den var kiselalgsskörd (Tabell 1). I tabellen ges även RMS-felet som ges av en modell som gissar snittet på skördparametern vid varje datapunkt. Alla modeller nådde ett lägre RMS-fel än vad som hade uppnåtts ifall de bara gissade snittet.

Tabell 1: Genomsnittligt lägsta RMS-fel för XGBoost-modell tränad att förutspå värde på en av tre skördeparametrar.

Skördparameter	Bästa modell-RMS	Snitt-RMS
Kiselalgsskörd	0,17	0,225
Kontamination	0,12	0,216
PAS	0,05	0,202

5 Diskussion

Diskussionen inleds av ett avsnitt om samhälleliga och etiska aspekter. Därpå följer motivering av metodval och tolkning av resultat. Diskussionen innehåller även förslag på en annan metod för datainsamling så att framtida studier är möjliga. Slutligen presenteras slutsatserna av arbetet.

5.1 Samhälleliga och etiska aspekter

Projektet omfattar enbart data från alger och involverar därför inga personuppgifter eller känslig information, vilket minimerar risken för intrång i integritet och autonomi. Samtidigt kan en ökad och mer effektiv algodling ge flera samhällsnyttor. Genom att utnyttja det näringsrika vattnet från fiskodlingar, exempelvis hos Smögenlax, skapas ett cirkulärt kretslopp där algerna tar upp näringen och sedan skickar tillbaka rent och syrerikt vatten till fiskarna. Detta främjar resurseffektivitet och hållbarhet. Dessutom kan storskalig odling av kiselalger bidra till att binda koldioxid och producera ämnen som kiseldioxid, vilka kan nyttjas inom flera sektorer för att utveckla klimatsmarta produkter och produktionsprocesser. Alger har dessutom ett brett användningsområde – från livsmedel och hälsoprodukter till industriella material och biobaserade bränslen, vilket kan stimulera innovation, skapa nya arbetstillfällen och stärka lokala näringar. I förlängningen kan sådana satsningar även bidra till en mer cirkulär ekonomi där naturresurser används på ett effektivt och långsiktigt hållbart sätt.

Trots ovan nämnda fördelar kräver en potentiell uppskalning av algodlingen noggranna överväganden av miljöeffekter. En storskalig produktion skulle kunna påverka ekosystem och resurstillgångar, exempelvis genom förändrad näringsbalans, pH-värde eller konkurrens med naturliga arter. Även om risken är begränsad när algerna odlas i ett växthus, är det ändå viktigt att följa upp dessa aspekter och säkerställa att odlingsanläggningarna är utformade för att minimera oönskad spridning eller påverkan på lokala ekosystem. Projektet är dessutom ett kandidatarbete med fokuserad omfattning, vilket innebär att resultaten främst är avsedda som förslag till ett enskilt företag och inte nödvändigtvis kommer att ge några omedelbara, storskaliga effekter på samhället. Eventuella större förändringar, såsom bred introduktion av nya odlingstekniker eller utökade produktionsvolymer, kräver också tid, resurser och engagemang från flera olika aktörer, för att säkerställa både ekonomisk och miljömässig hållbarhet.

En ytterligare etisk aspekt rör användningen av maskininlärning i projektet. Det finns många etiska frågeställningar förknippade med maskininlärning, och en hel rapport skulle kunna ägnas åt att diskutera dem; här ges därför endast en översikt. För det första är det centralt att granska vilken data som används och hur den har samlats in. I detta fall har data tillhandahållits av ett företag i syfte att analyseras, och samlades alltså inte in på ett oetiskt sätt. Data innehåller heller inga personuppgifter eller andra känsliga uppgifter eftersom det rör odlingsförhållanden för alger. För det andra är transparens kring modellens beslut avgörande för att möjliggöra granskning och förtroende. Detta har hanterats genom att beräkna SHAP-värden för de olika skördeparametrarna, vilket gör det möjligt att förstå vilka faktorer som påverkar prognosen och i vilken riktning.

5.2 Val av metod

I genomförandet av studien gjordes ett antal val som kan anses vara godtyckliga. Detta avsnitt ämnar att förklara dessa val och resonemanget bakom dem.

För modellträning behövde tidsseriedata konverteras till tabelldata, vilket gjordes genom att ta medelvärden över intervall. Att ta snittet över enskilda dagar ansågs vara för hög upplösning eftersom individuella odlingsbanor skördas var tredje vecka. För att minska antalet parametrar som modellen tränades över togs därför medelvärden över en eller tre veckor. Då krävdes bara att minst en dag i veckan som beräknades hade ett värde på en viss parameter för att snittet över veckan för denna parameter skulle bli nollskiljt, detta för att maximera det redan låga antalet datapunkter.

När XGBoost-modellerna skulle tränas användes flera olika modeller som tränades med olika kom-

binationer av variabler. Anledningen till detta var det relativt låga antalet skördestillfällen i data, vilket ökar risken för överanpassning om antalet variabler är för stort. Att använda fler variabler vid samma tillfälle skulle dessutom minska antalet datapunkter eftersom data för olika variabler saknas vid olika perioder och det räcker med att värdet för en variabel saknas för att en datapunkt ska exkluderas från träningsdata. Strategin att träna flera olika modeller på olika uppsättningar variabler valdes alltså för att utforska fler variablers påverkan på skörden utan att riskera överanpassning eller drastiskt begränsa antalet träningsdatapunkter.

Tidsvariabeln var alltid inkluderad i modellernas träningsdata när y-variabeln var PAS, detta eftersom det var tydligt från den tidigare dataanalysen att tid från start hade stor korrelation med renhetsgraden, samt att vissa variabler korrelerade relativt väl med tid. Vi ville då undvika möjligheten att modellen skulle fästa hög vikt vid någon variabel som egentligen inte har en stor effekt på renhetsgraden, utan bara korrelerar väl med tiden. Denna risk bestod trots att tidsvariabeln alltid inkluderades, men förhoppningen var att detta val åtminstone skulle ha minimerat den.

Vissa x-variabler inkluderades inte alls i modellträningen. Detta eftersom de antingen ansågs vara för lika andra variabler, eller hade värden vid för få datapunkter för att effektivt kunna träna modellen. Maximal och minimal temperatur över dygnet ansågs exempelvis vara för lika medelvärdet på temperaturen som användes istället. Samma gäller för ljusparametrarna, där enbart en valdes ut. Ljus % valdes eftersom den hade flest datapunkter.

5.3 Tolkning av resultat

Resultaten från studien indikerar att det inte går att identifiera någon tydligt avgörande faktor för tillväxten av kiselalger i den insamlade data. För de linjära analyserna visar sig detta i form av svaga korrelationer mellan tillväxt och de undersökta parametrarna. Inte heller för XGBoost-modellerna framträder någon enskild parameter enligt SHAP-värdena. Förbättring av RMS-felet jämfört med medelvärdet var i allmänhet marginell för XGBoost, vilket antyder att modellen haft begränsad förmåga att observera meningsfulla mönster ur data.

Denna bristande förmåga att identifiera faktorer kan sannolikt förklaras av flera metodologiska begränsningar i studien. Till viss del är det den relativt begränsade mängden skördar, och dels finns det variationer i odlingssystemet som inte fångas upp i den insamlade data. Detta då data kommer från systemtankar och utplacerade sensorer som inte mäter odlingsförhållandena med full representativitet, eftersom endast en tredjedel av modulerna skördas varje vecka. Innebörden blir att de abiotiska förhållandena kan skilja sig mellan olika delar av systemet, bland annat höjden från marken och avståndet till systemtanken.

Det begränsade antalet skördar reducerar också möjligheterna att identifiera tydliga och statistiska samband mellan odlingsparametrar och skörd. Speciellt försvåras möjligheten då många parametrar dessutom har visat sig samvariera över tid. Begränsningarna påverkar även modellernas möjlighet att observera mer komplexa samband, som potentiellt har en viktig inverkan på tillväxt och kontaminering i algodlingen. Tillsammans minskar dessa begränsningar studiens tillförlitlighet och resultat.

Vid analysen av grönalgskontaminationen framkommer ett tydligare samband med temperatur jämfört med övriga parametrar. Detta ses i såväl de linjära korrelationerna som i SHAP-värdena från XGBoost-modellen, där högre temperaturer sammanfaller med ökad mängd av grönalger. Att XGBoost i detta fall lyckades reducera RMS-felet i högre grad än för kiselalger stärker modellen något. Det bör dock understrykas att begränsningarna i data även påverkar på samma sätt och att flera parametrar korrelerar (Figur 3), vilket gör det svårt att med säkerhet fastställa orsak.

För PAS identifierades tiden som den mest avgörande faktorn i både korrelationsanalysen och XGBoost-modellen. Kontaminationshalten tenderar att öka över tid, vilket sannolikt har en negativ inverkan på skörden. Även RMS-felet minskade mer kraftigt än de andra skördeparametrarna som kan tyda på högre predikativ förmåga. Detta kan delvis vara en konsekvens av den lägre

variation i PAS jämfört med de två andra skördeparametrarna. Att tid är viktig för PAS väcker frågor om odlingsteknik, hantering av bekämpningsmedel och renlighet för odlingssystemet. Dessa aspekter bör undersökas vidare i framtida studier.

5.4 Förslag för framtida studier

För att möjliggöra en mer djupgående och tillförlitlig analys i framtida studier krävs en mer omfattande, systematisk och standardiserad datainsamling. Ett förbättringsförslag är att särskilja odlingsmodulerna från varandra i datainsamlingen. Detta görs genom att registrera skörd och parametrar per individuell modul eller genom att gruppera moduler utifrån gemensamma egenskaper, som exempelvis höjdnivå eller avstånd till systemtank. Det skulle bidra till att reducera den variation som i nuläget inte kan hanteras samtidigt som den totala datamängden hade ökat.

Större mängd data över tid hade skapat förutsättningar för att undersöka långsiktiga trender och tidsberoende effekter. Detta skulle exempelvis möjliggöra användning av mer avancerade maskininlärningsmodeller såsom long short-term memory (LSTM), vilka är särskilt lämpade för att analysera sekventiella data. Genom sådana modeller kan en djupare förståelse för algernas tillväxt utvecklas och i sin tur bidra till optimering av odlingsprocessen.

5.5 Slutsats

Studien undersökte sambandet mellan odlingsparametrar och både tillväxt av kiselalger samt kontaminering i ett modulärt algodlingssystem. Resultaten lyckades inte finna några tydliga faktorer som kan kopplas till kiselalgers tillväxt, vilket sannolikt beror på den begränsade datamängden och variationer inom odlingssystemet som inte fångats upp av den insamlade data. För kontaminering med grönalger framkom dock ett tydligare samband med temperatur och för PAS, procent användbar skörd, kunde tid identifieras som den viktigaste faktorn.

De begränsningarna som fanns i data minskade möjligheterna att identifiera samband och dra tillförlitliga slutsatser. För att framtida studier ska kunna ge mer djupgående insikter krävs en mer strukturerad och omfattad datainsamling. Detta skulle även möjliggöra användningen av mer avancerade analysmetoder och därigenom bidra till en bättre förståelse för de faktorer som påverkar både skörd och kontaminering i algodlingssystem.

Referenser

1. Encyclopædia Britannica. Diatoms [Fotografi]. 2016 Maj. Hämtad från: https://quest-eb-com.eu1.proxy.openathens.net/images/149_2099890. Hämtad: 2025-04-01
2. Wikipediaförfattare. Machine learning. Wikipedia, The Free Encyclopedia. 2025. Hämtad från: https://en.wikipedia.org/wiki/Machine_learning. Hämtad: 2025-02-07
3. Swedish Algae Factory. Swedish Algae Factory - a visionary algae team. Hämtad från: https://www.swedishalgaefactory.com/story/the-start. Hämtad: 2025-02-26
4. Swedish Algae Factory. One ingredient, multiple benefits. Hämtad från: https://www.algica.com/personal-care. Hämtad: 2025-02-26
5. Bandara TMWJ, Furlani M, Albinsson I, Wulff A och Mellander BE. Diatom Frustules Enhancing the Efficiency of Gel Polymer Electrolyte Based Dye-Sensitized Solar Cells with Multilayer Photoelectrodes. Nanoscale Advances 2019 Dec; 2:199–209. DOI: 10.1039/c9na00679f
6. Swedish Algae Factory. The circular process. Hämtad från: https://www.swedishalgaefactory.com/circularity/circularity-in-practice. Hämtad: 2025-02-26
7. Kumar D och Singh B. Algal Biorefinery: An Integrated Approach for Sustainable Biodiesel Production. Biomass and Bioenergy 2019; 131. DOI: 10.1016/j.biombioe.2019.105398. Hämtad från: https://www.sciencedirect.com/science/article/pii/S0961953419303285.
8. Madeira MS, Cardoso C, Lopes PA, Coelho D, Afonso C, Bandarra NM och Prates JAM. Microalgae as Feed Ingredients for Livestock Production and Meat Quality: A Review. Livestock Science 2017; 205:111–21. DOI: 10.1016/j.livsci.2017.09.020. Hämtad från: https://www.sciencedirect.com/science/article/pii/S1871141317302858.
9. Lauritano C, Andersen JH, Hansen E, Albrigtsen M, Escalera L, Esposito F, Helland K, Hanssen KØ, Romano G och Ianora A. Bioactivity Screening of Microalgae for Antioxidant, Anti-Inflammatory, Anticancer, Anti-Diabetes, and Antibacterial Activities. Frontiers in Marine Science 2016; 3. DOI: 10.3389/fmars.2016.00068. Hämtad från: https://www.frontiersin.org/journals/marine-science/articles/10.3389/fmars.2016.00068.
10. Chatzilygeroudis K, Hatzilygeroudis I och Perikos I. Machine Learning Basics. *Intelligent Computing for Interactive System Design: Statistics, Digital Signal Processing, and Machine Learning in Practice*. 1. utg. New York, NY, USA: Association for Computing Machinery, 2021 :143–93. Hämtad från: https://doi.org/10.1145/3447404.3447414.
11. Jovanović A, Krstić A, Vujnović S och Durović Ž. On Multivariate Linear Regression Applications. *Proceedings of the 2024 11th International Conference on Electrical, Electronic and Computing Engineering (IcETRAN)*. IEEE, 2024. DOI: 10.1109/IcETRAN62308.2024.10645121. Hämtad från: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85204070664&doi=10.1109%2fIcETRAN62308.2024.10645121&partnerID=40&md5=c8d88777395f81b26a3b83bf2caeea22.
12. Hyndman RJ och Koehler AB. Another Look at Measures of Forecast Accuracy. International Journal of Forecasting 2006; 22:679–88. DOI: 10.1016/j.ijforecast.2006.03.001. Hämtad från: https://www.sciencedirect.com/science/article/pii/S0169207006000239.
13. Bentéjac C, Csörgő A och Martínez-Muñoz G. A Comparative Analysis of XGBoost. Artificial Intelligence Review 2021; 54:1937–67. DOI: 10.1007/s10462-020-09896-5. Hämtad från: https://doi.org/10.1007/s10462-020-09896-5.
14. Chen T och Guestrin C. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016 :785–94. DOI: 10.1145/2939672.2939785. Hämtad från: https://doi.org/10.1145/2939672.2939785.
15. Yang L och Shami A. On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice. Neurocomputing 2020; 415:295–316. DOI: 10.1016/j.neucom.2020.07.061. Hämtad från: https://www.sciencedirect.com/science/article/pii/S0925231220311693.

16. Bergstra J och Bengio Y. Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research 2012; 13:281–305. Hämtad från: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84857855190&partnerID=40&md5=fe76c9d1e6e5375afab7760adbe1ae4f.

17. Lundberg S och Lee SI. A Unified Approach to Interpreting Model Predictions. 2017. arXiv: 1705.07874 [cs.AI]. Hämtad från: https://arxiv.org/abs/1705.07874.

18. Armbrust EV. The life of diatoms in the world's oceans. Nature 2009; 459:185–92. DOI: 10.1038/nature08057. Hämtad från: https://doi.org/10.1038/nature08057.

19. De Tommasi E, Gielis J och Rogato A. Diatom Frustule Morphogenesis and Function: A Multidisciplinary Survey. Marine Genomics 2017; 35:1–18. DOI: 10.1016/j.margen.2017.07.001. Hämtad från: https://www.sciencedirect.com/science/article/pii/S1874778717301265.

Användande av AI

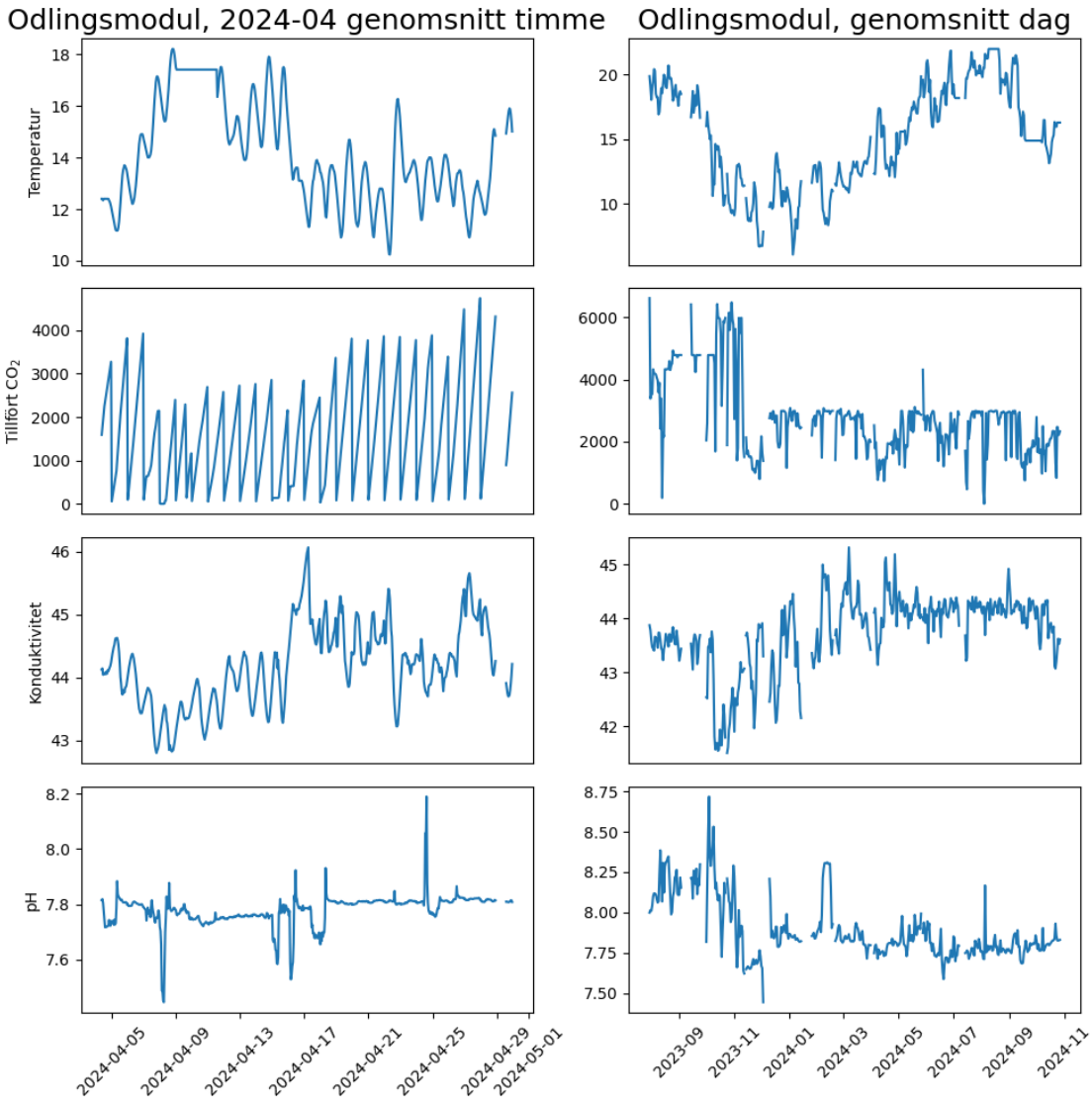
I det här arbetet har generativ AI använts till ett fåtal uppgifter. Bland annat har AI använts för att formatera referenser korrekt i biblatex-format och för att utvärdera stycken i texten. Alla förslag som givits har granskats noggrant, och några har arbetats in i rapporten. AI har också använts för att skriva abstract, där den fick översätta sammanfattningen till engelska. Utöver detta har AI också använts för att underlätta vid kodning, bland annat för att förklara felmeddelanden, ge förslag på metoder för att utföra specifika operationer och för att dubbelkolla att ett stycke kod verkligen gör det man vill att den ska göra. Syftet med användandet var i grunden alltid att spara tid på uppgifter som inte ansågs vara centrala för projektet. Alla problem som löstes med hjälp av AI hade kunnat lösas utan dess hjälp men hade inneburit tidskrävande och simpelt arbete som att exempelvis läsa dokumentationen för funktioner som används i koden.

A Kompletterande resultat och figurer

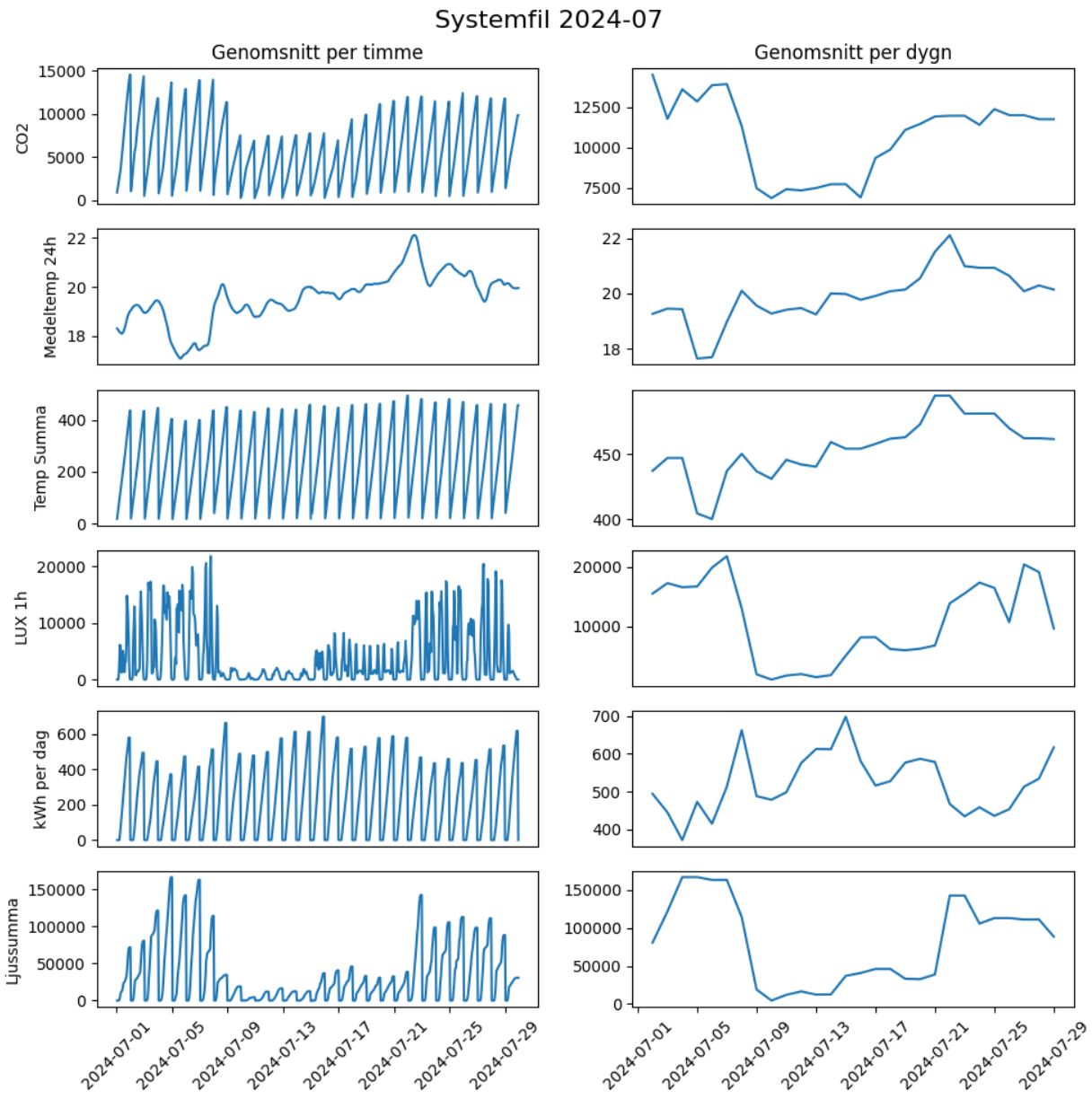
Detta appendix innehåller figurer som visar data från studiens olika dataset, data efter databehandling samt resultat av linjär regression.

A.1 Datavisualisering

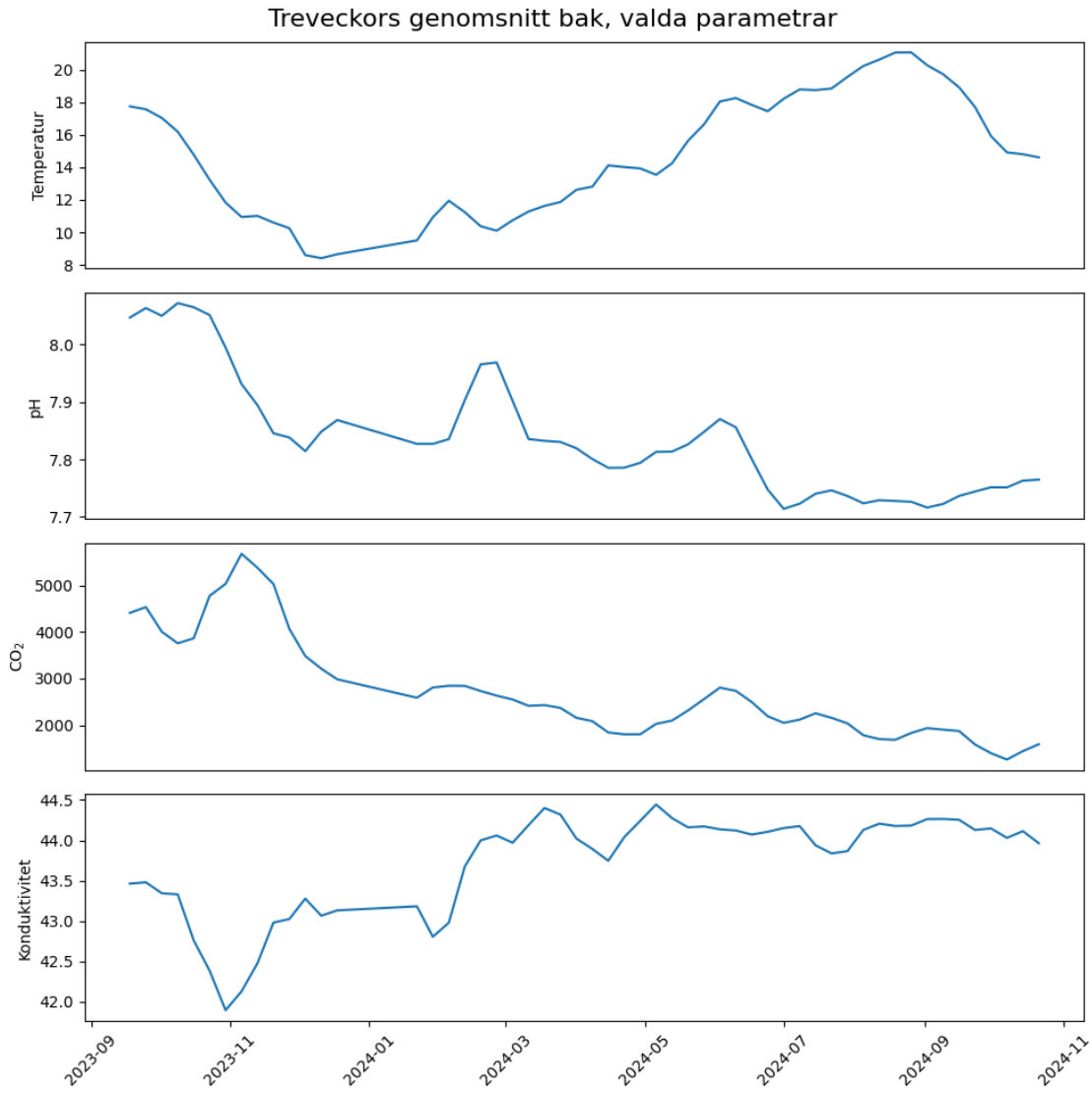
Figur 1: Data från parametrarna i datasetet odlingsresultat. Parametrarna är ljus %, ljus i watt, korrigerad kWh, ljuseffekt kWh, tot ljus kWh, ljussumma sol LUX, CO_2, temp Vmedel, temp Vmax, temp Vmin och temp summa. Data visas för olika perioder beroende på vilka intervall mätningarna gjorts på. Vissa perioder saknar eller har konstanta värden troligen kopplade till felaktiga mätningar.



Figur 2: Data från odlingsmodulernas systemtank, OM-A och OM-B tillsammans. Parametrarna som visas är temperatur, tillfört CO_2, konduktivitet och pH. Till vänster syns systemtankens parametrarna för april 2024 taget som genomsnittet för varje hel timme. Till höger syns hela periodens data med taget dygnsgenomsnitt istället. Observera att vissa intervall saknar data och vissa intervall har konstanta värden på vissa parametrar troligen relaterat till planerade eller okända fel på mätarna.

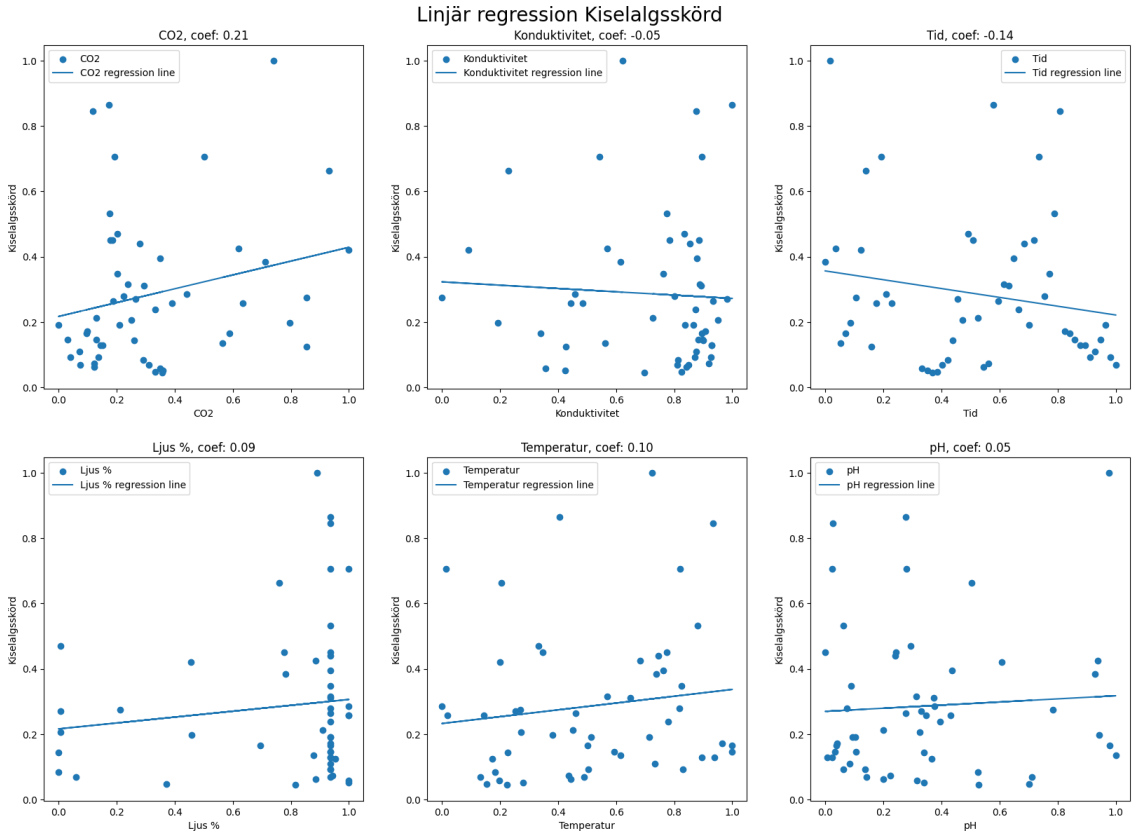


Figur 3: Exempeldata från systemfilen på de olika parametrarna under juli 2024. Till vänster syns genomsnittet per timme medans till höger syns genomsnittet per dag.

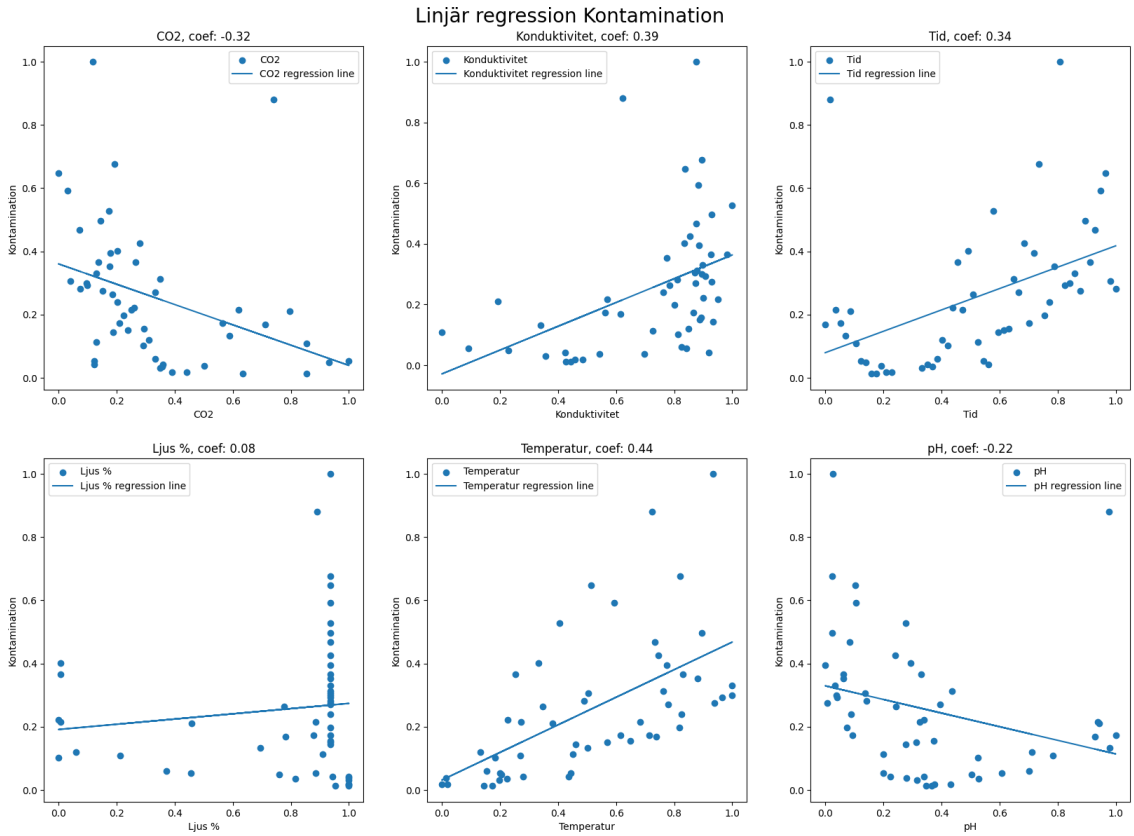


Figur 4: Data för parametrarna temperatur, pH, CO_2 och konduktivitet från odlingsmodulerna samt ljus % från filen odlingsresultat. Data sträcker sig lite längre än ett år och är vid varje punkt taget som tre veckors genomsnitt bakåt.

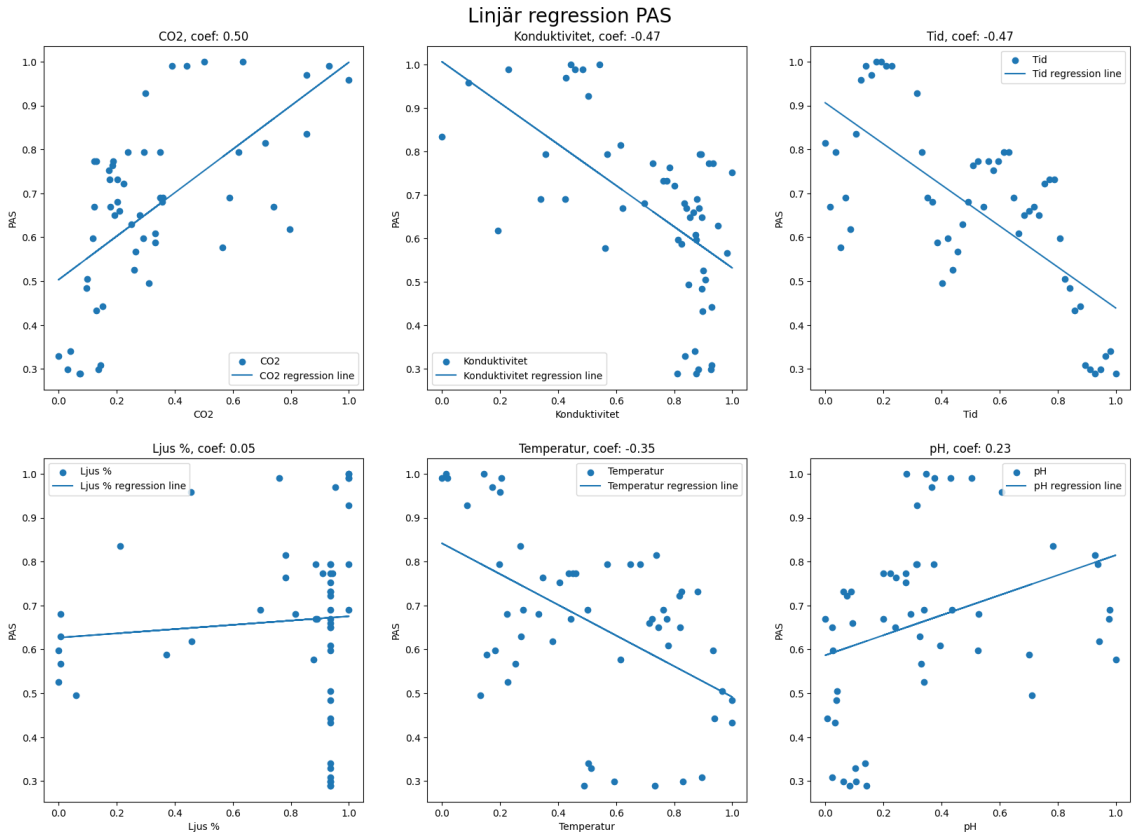
A.2 Figurer för linjär regression



Figur 5: Linjer är den linjära regressionen mellan kiselalgsskörd och valda parametrar temperatur, pH, konduktivitet och CO_2 från odlingsmodul samt ljus % från odlingsresultat. Data för beräkning av linjen är tagen som treveckors medelvärde bakåt i tiden. Tid-parametern är satt som datumet på den måndag efter de tre veckorna, det vill säga dagen för skörd.



Figur 6: Linjer är den linjära regressionen mellan korrelation och valda parametrar temperatur, pH, konduktivitet och CO_2 från odlingsmodul samt ljus % från odlingsresultat. Data för beräkning av linjen är tagen som treveckors medelvärde bakåt i tiden. Tid-parametern är satt som datumet på den måndag efter de tre veckorna, det vill säga dagen för skörd.



Figur 7: Linjer är den linjära regressionen mellan PAS och valda parametrar temperatur, pH, konduktivitet och CO_2 från odlingsmodul samt ljus % från odlingsresultat. Data för beräkning av linjen är tagen som treveckors medelvärde bakåt i tiden. Tid-parametern är satt som datumet på den måndag efter de tre veckorna, det vill säga dagen för skörd.

B Källkod

B.1 Odlingsmodul databehandling

Följande källkod är till för att hantera odlingsmodulerna. Koden i fråga är för modul A men hantering av modul B körs identiskt med andra indatafiler.

```
# %%
# Select files in folder data and filter for containing OM-A
import os
import re

files = os.listdir('data2/Rådata/Rådata')
files = [f for f in files if re.search('OM-A', f)]
print(files)

# %%
import pandas as pd
dfs = []
for f in files:
    df = pd.read_csv('data2/Rådata/Rådata/' + f, skipfooter=9, sep=';', ...
        engine='python', index_col=False)
    if f == 'OM-A_2024-05-05.csv':
        df.drop('Unnamed: 9', axis=1, inplace=True)
        df.drop('Unnamed: 10', axis=1, inplace=True)
        df['pH - A'] = df['pH - A'].str.replace(',', '.').astype(float)
        df['Tillfört CO2   - A'] = df['Tillfört CO2   - A'].str.replace(',', ...
            '.').astype(float)
        df['Konduktivitet'] = df['Konduktivitet'].str.replace(',', '.').astype(float)
        df['Temp A'] = df['Temp A'].str.replace(',', '.').astype(float)
    df.drop('Unnamed: 8', axis=1, inplace=True)
    dfs.append(df)
# combine dataframes
df_raw = pd.concat(dfs)
df_raw.tail()

# %%
df_raw.isna().sum()

# %%
# read data from files into pandas dataframes
import numpy as np
df = df_raw.copy()
df.dropna(inplace=True)
df['Date'] = pd.to_datetime(df['Date'], dayfirst=True)
df.loc[df['Konduktivitet'] < 41.5,'Konduktivitet'] = np.nan
df['Konduktivitet'] = df['Konduktivitet'].interpolate(method='linear')
df.loc[df['Temp A'] < 5,'Temp A'] = np.nan
df['Temp A'] = df['Temp A'].interpolate(method='linear')
df.loc[df['pH - A'] < 7,'pH - A'] = np.nan
df.loc[df['pH - A'] > 9,'pH - A'] = np.nan
df['pH - A'] = df['pH - A'].interpolate(method='linear')
df

# %%
df.info()

# %%
df.describe()

# %%
# 1T = 1 minute, 1H = 1 hour, 1D = 1 day, 1W = 1 week, 1M = 1 month, 1Y = 1 year
df.set_index('Date', inplace=True)
resampled = df.resample('60T').mean()
resampled.reset_index(inplace=True)

resampled24 = df.resample('1D').mean()
resampled24.reset_index(inplace=True)
```

```
df.reset_index(inplace=True)
resampled

# %%
import matplotlib.pyplot as plt
# plot all variables against time in one seperate subplots columns, 'Volym ...
    systemtank A', 'pH - A', 'Antal aktiva banor - A','Tillfört CO2   - A', ...
    'Konduktivitet', 'Temp A', 'Skördsump_A'
fig, axs = plt.subplots(4, 2, figsize=(10, 10))
axs[0,0].set_title('Odlingsmodul, 2024-04 genomsnitt timme', fontsize=18)
axs[0,1].set_title('Odlingsmodul, genomsnitt dag', fontsize=18)
df_fig = resampled24.copy()
axs[0,1].plot(df_fig['Date'].values, df_fig['Temp A'].values)
#axs[0,1].set_ylabel('CO2')
axs[1,1].plot(df_fig['Date'].values, df_fig['Tillfört CO2   - A'].values)
#axs[1,1].set_ylabel('Medeltemp 24h')
axs[2,1].plot(df_fig['Date'].values, df_fig['Konduktivitet'].values)
#axs[2,1].set_ylabel('Temp Summa')
axs[3,1].plot(df_fig['Date'].values, df_fig['pH - A'].values)
#axs[3,1].set_ylabel('LUX 1h')
#axs[4,1].plot(df_fig['Date'].values, df_fig['kWh per dag'].values)
#axs[4,1].set_ylabel('kWh per dag')
#axs[5,1].plot(df_fig['Date'].values, df_fig['Ljussumma'].values) # .interpolate()
#axs[5,1].set_ylabel('Ljussumma')
axs[0,1].get_xaxis().set_visible(False)
axs[1,1].get_xaxis().set_visible(False)
axs[2,1].get_xaxis().set_visible(False)
#axs[3,1].get_xaxis().set_visible(False)
#axs[4,1].get_xaxis().set_visible(False)
axs[3,1].tick_params(axis='x', rotation=45)

df_fig = resampled.query('Date > "2024-04-01" and Date < "2024-04-30"')
axs[0,0].plot(df_fig['Date'].values, df_fig['Temp A'].values)
axs[0,0].set_ylabel('Temperatur')
axs[1,0].plot(df_fig['Date'].values, df_fig['Tillfört CO2   - A'].values)
axs[1,0].set_ylabel('Tillfört CO$_2$')
axs[2,0].plot(df_fig['Date'].values, df_fig['Konduktivitet'].values)
axs[2,0].set_ylabel('Konduktivitet')
axs[3,0].plot(df_fig['Date'].values, df_fig['pH - A'].values)
axs[3,0].set_ylabel('pH')
axs[0,0].get_xaxis().set_visible(False)
axs[1,0].get_xaxis().set_visible(False)
axs[2,0].get_xaxis().set_visible(False)
axs[3,0].tick_params(axis='x', rotation=45)

#axs[0,0].set_xticklabels(df_fig['Date'].values, rotation=45)
#axs[0,1].set_xticklabels(df_fig['Date'].values, rotation=45)

plt.tight_layout()
plt.savefig('finalfigs/OM-A_avg_hour_and_day.png')
plt.show()

# %%
resampled.to_csv('output/A_resampled.csv', index=False)


# %%
import pandas as pd
import os

df = pd.read_csv('output/A_resampled.csv')

df['Date'] = pd.to_datetime(df['Date'])

# %%
# aggregate mean,max,min,std temp of every day
daily_agg_t = df.resample('D', on='Date').agg({
    'Temp A': ['mean', 'max', 'min', 'std']
}).reset_index()
```

```
daily_agg_t.columns = ['Date', 'Temp_mean', 'Temp_max', 'Temp_min', 'Temp_std']
daily_agg_t

# %%
# do same for ph
daily_agg_ph = df.resample('D', on='Date').agg({
    'pH - A': ['mean', 'max', 'min', 'std']
}).reset_index()

daily_agg_ph.columns = ['Date', 'pH_mean', 'pH_max', 'pH_min', 'pH_std']
daily_agg_ph

# %%
daily_agg_k = df.resample('D', on='Date').agg({
    'Konduktivitet': ['mean', 'max', 'min', 'std']
}).reset_index()

daily_agg_k.columns = ['Date', 'Conductivity_mean', 'Conductivity_max', ...
    'Conductivity_min', 'Conductivity_std']
daily_agg_k

# %%
daily_agg_c = df.resample('D', on='Date').agg({
    'Tillfört CO2   - A': ['mean', 'max', 'min', 'std']
}).reset_index()

daily_agg_c.columns = ['Date', 'CO2_mean', 'CO2_max', 'CO2_min', 'CO2_std']
daily_agg_c

# %%
# combine all on date
daily_agg = pd.merge(daily_agg_t, daily_agg_ph, on='Date')
daily_agg = pd.merge(daily_agg, daily_agg_k, on='Date')
daily_agg = pd.merge(daily_agg, daily_agg_c, on='Date')

# Interpolate linear for missing week
daily_agg.loc[(daily_agg['Date'] ≥ '2024-01-15') & (daily_agg['Date'] ≤ ...
    '2024-01-25'), daily_agg.columns[1:]] = None
# interpolate
daily_agg = daily_agg.interpolate(method='linear')

# %%
daily_agg.to_csv('output/A_daily_agg.csv', index=False)

# %%
import matplotlib.pyplot as plt
plt.plot(daily_agg['Date'].values, daily_agg['Temp_mean'].values)
# rotate axis
plt.xticks(rotation=45)
```

B.2 Systemfil databehandling

Följande källkod behandlar systemfilens databehandling enligt metodbeskrivingen.

```
    # %%
# Select files in folder data and filter for containing OM-A
import os
import re

files = os.listdir('data2/Rådata')
files = [f for f in files if re.search('Gustav', f)]
print(files)

# %%
import pandas as pd
dfs = []
b = 0
```

```
    for f in files:
        df = pd.read_csv('data2/Rådata/' + f, skipfooter=9, sep=';', engine='python', ...
            index_col=False)
        if 'Ljussumma' not in df.columns:
            df.rename({'Unnamed: 6': 'Ljussumma'}, axis=1, inplace=True)
        if 'Unnamed: 7' in df.columns:
            df.drop('Unnamed: 7', axis=1, inplace=True)
        dfs.append(df)
    # combine dataframes
    df_raw = pd.concat(dfs)
    df_raw.head()

    # %%
    # read data from files into pandas dataframes
    import numpy as np
    df = df_raw.copy()
    #df.dropna(inplace=True)
    df['Date'] = pd.to_datetime(df['Date'], dayfirst=True)
    #df.loc[df['LUX 1h'] > 5000,'LUX 1h'] = np.nan
    #df['LUX 1h'] = df['LUX 1h'].interpolate(method='linear')
    df

    # %%
    import matplotlib.pyplot as plt
    # plot all variables against time in one seperate subplots columns, 'Volym ...
        systemtank A', 'pH - A', 'Antal aktiva banor - A','Tillfört CO2   - A', ...
        'Konduktivitet', 'Temp A', 'Skördsump_A'
    fig, axs = plt.subplots(3, 2, figsize=(10, 10))
    df_fig = df#.query('Date > "2024-07-01" and Date < "2024-07-14"')
    axs[0, 0].plot(df_fig['Date'].values, df_fig['CO2'].values)
    axs[0, 0].set_title('CO2')
    axs[0, 1].plot(df_fig['Date'].values, df_fig['Medeltemp 24h'].values)
    axs[0, 1].set_title('Medeltemp 24h')
    axs[1, 0].plot(df_fig['Date'].values, df_fig['TEMP SUMMA'].values)
    axs[1, 0].set_title('TEMP SUMMA')
    axs[1, 1].plot(df_fig['Date'].values, df_fig['LUX 1h'].values)
    axs[1, 1].set_title('LUX 1h')
    axs[2, 0].plot(df_fig['Date'].values, df_fig['kWh per dag'].values)
    axs[2, 0].set_title('kWh per dag')
    axs[2, 1].plot(df_fig['Date'].values, df_fig['Ljussumma'].values) # .interpolate()
    axs[2, 1].set_title('Ljussumma')
    #axs[3, 0].plot(df['Date'].values, df['Skördsump_A'].values)
    #axs[3, 0].set_title('Skördsump_A')
    #axs[3, 1].plot(df['Date'].values, df['Skördsump_A'].values)
    #axs[3, 1].set_title('Skördsump_A')
    # axis ticks x only first and last
    #axs[0,0].get_xaxis().set_visible(False)
    #axs[0,1].get_xaxis().set_visible(False)
    #axs[1,0].get_xaxis().set_visible(False)
    #axs[1,1].get_xaxis().set_visible(False)
    #axs[2,0].get_xaxis().set_visible(False)
    #axs[2,1].get_xaxis().set_visible(False)
    # rotate x-axis labels
    axs[2,0].tick_params(axis='x', rotation=45)
    axs[2,1].tick_params(axis='x', rotation=45)
    plt.tight_layout()
    plt.savefig('figures/Gustav_all_data.png')
    plt.show()

    # %%
    # 1T = 1 minute, 1H = 1 hour, 1D = 1 day, 1W = 1 week, 1M = 1 month, 1Y = 1 year
    df.set_index('Date', inplace=True)
    resampled = df.resample('1H').max()
    resampled.reset_index(inplace=True)
    df.reset_index(inplace=True)
    resampled

    # %%
    import matplotlib.pyplot as plt
```

```
    # plot all variables against time in one seperate subplots columns, 'Volym ...
        systemtank A', 'pH - A', 'Antal aktiva banor - A','Tillfört CO2   - A', ...
        'Konduktivitet', 'Temp A', 'Skördsump_A'
    fig, axs = plt.subplots(6, 1, figsize=(10, 10))
    fig.suptitle('Systemfil 2024-07, genomsnitt per dag', fontsize=16)
    df_fig = resampled.query('Date > "2024-07-01" and Date < "2024-07-30"')
    axs[0].plot(df_fig['Date'].values, df_fig['CO2'].values)
    axs[0].set_ylabel('CO$_2$')
    axs[1].plot(df_fig['Date'].values, df_fig['Medeltemp 24h'].values)
    axs[1].set_ylabel('Medeltemp 24h')
    axs[2].plot(df_fig['Date'].values, df_fig['TEMP SUMMA'].values)
    axs[2].set_ylabel('Temp Summa')
    axs[3].plot(df_fig['Date'].values, df_fig['LUX 1h'].values)
    axs[3].set_ylabel('LUX 1h')
    axs[4].plot(df_fig['Date'].values, df_fig['kWh per dag'].values)
    axs[4].set_ylabel('kWh per dag')
    axs[5].plot(df_fig['Date'].values, df_fig['Ljussumma'].values) # .interpolate()
    axs[5].set_ylabel('Ljussumma')
    #axs[3, 0].plot(df['Date'].values, df['Skördsump_A'].values)
    #axs[3, 0].set_title('Skördsump_A')
    #axs[3, 1].plot(df['Date'].values, df['Skördsump_A'].values)
    #axs[3, 1].set_title('Skördsump_A')
    # axis ticks x only first and last
    #axs[0,0].get_xaxis().set_visible(False)
    #axs[0,1].get_xaxis().set_visible(False)
    #axs[1,0].get_xaxis().set_visible(False)
    #axs[1,1].get_xaxis().set_visible(False)
    #axs[2,0].get_xaxis().set_visible(False)
    #axs[2,1].get_xaxis().set_visible(False)
    axs[0].get_xaxis().set_visible(False)
    axs[1].get_xaxis().set_visible(False)
    axs[2].get_xaxis().set_visible(False)
    axs[3].get_xaxis().set_visible(False)
    axs[4].get_xaxis().set_visible(False)
    # rotate x-axis labels
    axs[5].tick_params(axis='x')
    #axs[2,1].tick_params(axis='x', rotation=45)
    plt.tight_layout()
    plt.savefig('figures/Gustav_avg_per_day_2024_07.png')
    plt.show()

    # %%
    # 1T = 1 minute, 1H = 1 hour, 1D = 1 day, 1W = 1 week, 1M = 1 month, 1Y = 1 year
    df.set_index('Date', inplace=True)
    resampled24 = df.resample('24H').max()
    resampled24.reset_index(inplace=True)
    df.reset_index(inplace=True)
    resampled24

    # %%
    import matplotlib.pyplot as plt
    # plot all variables against time in one seperate subplots columns, 'Volym ...
        systemtank A', 'pH - A', 'Antal aktiva banor - A','Tillfört CO2   - A', ...
        'Konduktivitet', 'Temp A', 'Skördsump_A'
    fig, axs = plt.subplots(6, 2, figsize=(10, 10))
    fig.suptitle('Systemfil 2024-07', fontsize=16)
    axs[0,0].set_title('Genomsnitt per timme', fontsize=12)
    axs[0,1].set_title('Genomsnitt per dygn', fontsize=12)
    df_fig = resampled24.query('Date > "2024-07-01" and Date < "2024-07-30"')
    axs[0,1].plot(df_fig['Date'].values, df_fig['CO2'].values)
    #axs[0,1].set_ylabel('CO2')
    axs[1,1].plot(df_fig['Date'].values, df_fig['Medeltemp 24h'].values)
    #axs[1,1].set_ylabel('Medeltemp 24h')
    axs[2,1].plot(df_fig['Date'].values, df_fig['TEMP SUMMA'].values)
    #axs[2,1].set_ylabel('Temp Summa')
    axs[3,1].plot(df_fig['Date'].values, df_fig['LUX 1h'].values)
    #axs[3,1].set_ylabel('LUX 1h')
    axs[4,1].plot(df_fig['Date'].values, df_fig['kWh per dag'].values)
    #axs[4,1].set_ylabel('kWh per dag')
```

```
axs[5,1].plot(df_fig['Date'].values, df_fig['Ljussumma'].values) # .interpolate()
#axs[5,1].set_ylabel('Ljussumma')
axs[0,1].get_xaxis().set_visible(False)
axs[1,1].get_xaxis().set_visible(False)
axs[2,1].get_xaxis().set_visible(False)
axs[3,1].get_xaxis().set_visible(False)
axs[4,1].get_xaxis().set_visible(False)
axs[5,1].tick_params(axis='x', rotation=45)

df_fig = resampled.query('Date > "2024-07-01" and Date < "2024-07-30"')
axs[0,0].plot(df_fig['Date'].values, df_fig['CO2'].values)
axs[0,0].set_ylabel('CO2')
axs[1,0].plot(df_fig['Date'].values, df_fig['Medeltemp 24h'].values)
axs[1,0].set_ylabel('Medeltemp 24h')
axs[2,0].plot(df_fig['Date'].values, df_fig['TEMP SUMMA'].values)
axs[2,0].set_ylabel('Temp Summa')
axs[3,0].plot(df_fig['Date'].values, df_fig['LUX 1h'].values)
axs[3,0].set_ylabel('LUX 1h')
axs[4,0].plot(df_fig['Date'].values, df_fig['kWh per dag'].values)
axs[4,0].set_ylabel('kWh per dag')
axs[5,0].plot(df_fig['Date'].values, df_fig['Ljussumma'].values) # .interpolate()
axs[5,0].set_ylabel('Ljussumma')
axs[0,0].get_xaxis().set_visible(False)
axs[1,0].get_xaxis().set_visible(False)
axs[2,0].get_xaxis().set_visible(False)
axs[3,0].get_xaxis().set_visible(False)
axs[4,0].get_xaxis().set_visible(False)
axs[5,0].tick_params(axis='x', rotation=45)

#axs[0,0].set_xticklabels(df_fig['Date'].values, rotation=45)
#axs[0,1].set_xticklabels(df_fig['Date'].values, rotation=45)

plt.tight_layout()
plt.savefig('finalfigs/Gustav_avg_hour_and_day_2024_07.png')
plt.show()

# %%
resampled.to_csv('output/Gustav_resampled.csv', index=False)

# %%
resampled
```

B.3 Odlingsresultat databehandling

Nedan följer pythonkod för databehandling och visualisering av odlingsresultat data.

```
    # %%
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Load the data
df_raw = pd.read_csv('data2/Rådata/Odlingsresultat.csv')
df_raw.rename(columns={'datum': 'Date'}, inplace=True)
df_raw

# %%
df = df_raw.copy()
df['Date'] = pd.to_datetime(df['Date'], dayfirst=False)
df['skörd (L pelletvolym)'] = df['skörd (L pelletvolym)'].str.replace(',', ...
    '.').astype(float)
df['Ljus %'] = df['Ljus %'].str.replace('%','').astype(float)
df['Ljus i watt'] = df['Ljus i watt'].str.replace(',', '.').astype(float)
df['Mätt ljuseffekt kWh per dag'] = df['Mätt ljuseffekt kWh per ...
    dag'].str.replace(',', '.').astype(float)
df['tot Ljus kWh'] = df['tot Ljus kWh'].str.replace(',', '.').astype(float)
df['Ljussumma Sol LUX'] = df['Ljussumma Sol LUX'].str.replace(',', '.').astype(float)
```

```
df['CO2'] = df['CO2'].str.replace(',', '.').astype(float)
df['temp Vmedel'] = df['temp Vmedel'].str.replace(',', '.').astype(float)
df['temp Vmax'] = df['temp Vmax'].str.replace(',', '.').astype(float)
df['temp Vmin'] = df['temp Vmin'].str.replace(',', '.').astype(float)
df['TEMP SUMMA'] = df['TEMP SUMMA'].str.replace(',', '.').astype(float)
df['Kontamination'] = df['Kontamination'].str.replace(',', '.').astype(float)
df['Microbiol'] = df['Microbiol'].str.replace('%', '').astype(float)

# %%
df.info()

# %%
df.describe()

# %%
df.columns

# %%
fig, ax = plt.subplots(6,2, figsize=(15,15))
fig.suptitle('Odlingsresultat parametrar', fontsize=24, y=1)
ax[0,0].plot(df['Date'], df['Ljus %'])
ax[0,0].set_ylabel('Ljus %', fontsize=16)
ax[1,0].plot(df['Date'], df['Ljus i watt'])
ax[1,0].set_ylabel('Ljus i watt', fontsize=16)
ax[1,1].plot(df['Date'], df['Beräknad korrigerad kWh per dygn'])
ax[1,1].set_ylabel('Korrigerad kWh', fontsize=16)
ax[3,0].plot(df['Date'], df['Mätt ljuseffekt kWh per dag'])
ax[3,0].set_ylabel('Ljuseffekt kWh', fontsize=16)
ax[4,0].plot(df['Date'], df['tot Ljus kWh'])
ax[4,0].set_ylabel('Tot Ljus kWh', fontsize=16)
ax[5,0].plot(df['Date'], df['TEMP SUMMA'])
ax[5,0].set_ylabel('Temp Summa', fontsize=16)
ax[0,1].plot(df['Date'], df['Ljussumma Sol LUX'])
ax[0,1].set_ylabel('Ljussumma Sol LUX', fontsize=16)
ax[2,0].plot(df['Date'], df['CO2'])
ax[2,0].set_ylabel('CO$_2$', fontsize=16)
ax[2,1].plot(df['Date'], df['temp Vmedel'])
ax[2,1].set_ylabel('temp Vmedel', fontsize=16)
ax[3,1].plot(df['Date'], df['temp Vmax'])
ax[3,1].set_ylabel('temp Vmax', fontsize=16)
ax[4,1].plot(df['Date'], df['temp Vmin'])
ax[4,1].set_ylabel('temp Vmin', fontsize=16)
ax[5,1].axis('off')
plt.tight_layout()
plt.savefig('finalfigs/odlingsresultat_params.png', dpi=300, bbox_inches='tight')
plt.show()

# %%
fig, ax = plt.subplots(3,1, figsize=(15,15))
fig.suptitle('Odlingsresultat skördeparametrar', fontsize=20, y=1)
ax[0].plot(df.query('dag == "Måndag"')['Date'], df.query('dag == ...
    "Måndag"')['Diatom skörd'])
ax[0].set_ylabel('Kiselalgsskörd', fontsize=16)
ax[1].plot(df.query('dag == "Måndag"')['Date'], df.query('dag == ...
    "Måndag"')['Kontamination'])
ax[1].set_ylabel('Kontamination', fontsize=16)
ax[2].plot(df.query('dag == "Måndag"')['Date'], df.query('dag == ...
    "Måndag"')['Microbiol'])
ax[2].set_ylabel('PAS', fontsize=16)
#ax[7,0].plot(df.query('dag == "Måndag"')['Date'], df.query('dag == ...
    "Måndag"')['skörd (L pelletvolym)'])
#ax[7,0].set_title('skörd (L pelletvolym)')
#ax[7,1].plot(df.query('dag == "Måndag"')['Date'], df.query('dag == ...
    "Måndag"')['Skörd ml'])
#ax[7,1].set_title('Skörd ml')
plt.tight_layout()
plt.savefig('finalfigs/odlingsresultat_skörd.png', dpi=300, bbox_inches='tight')
plt.show()
```

```
# %%
df.to_csv('output/odlingsresultat_cleaned.csv', index=False)
```

B.4 Sammanställning och analys

Följande kod innehåller både sammanställningen av treveckors medelvärden och datavisualisering av dem, samt enklare analys med XGBoost, linjär regression och korrelationer.

```
# %%
# combine A_daily_agg and odlingsresultat_cleaned

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

df_A_daily_agg = pd.read_csv('output/A_daily_agg.csv')
df_B_daily_agg = pd.read_csv('output/B_daily_agg.csv')
df_odlingsresultat_cleaned = pd.read_csv('output/odlingsresultat_cleaned.csv')

df_A_daily_agg['Date'] = pd.to_datetime(df_A_daily_agg['Date'])
df_B_daily_agg['Date'] = pd.to_datetime(df_B_daily_agg['Date'])
df_odlingsresultat_cleaned['Date'] = ...
    pd.to_datetime(df_odlingsresultat_cleaned['Date'])

df_gustav = pd.read_csv('output/Gustav_resampled.csv')
df_gustav['Date'] = pd.to_datetime(df_gustav['Date'])

#select from 2023-07-30
df_A_daily_agg = df_A_daily_agg[df_A_daily_agg['Date'] ≥ '2023-07-30']
df_B_daily_agg = df_B_daily_agg[df_B_daily_agg['Date'] ≥ '2023-07-30']
df_agg = (df_A_daily_agg.set_index('Date') + df_B_daily_agg.set_index('Date')) / 2
df_agg.reset_index(inplace=True)
df = pd.merge(df_agg, df_odlingsresultat_cleaned, on='Date', how='outer')
df = pd.merge(df, df_gustav, on='Date', how='outer')


# %%
df.columns

# %%
# remove nan dates
df = df.dropna(subset=['Date'])
# remove dates after date 1+/28/2024
df = df[df['Date'] < '2024-10-28']
df.sort_index(inplace=True)
df

# %%
df[df['Date'].dt.weekday == 0]

# %%
df['Year'] = df['Date'].dt.year
df_B_daily_agg['Year'] = df_B_daily_agg['Date'].dt.year

# %%
df[df['Date'].dt.weekday == 0].loc[:, ['Date', 'Year', 'vecka', 'Diatom skörd', ...
    'skörd (L pelletvolym)', 'Kontamination', 'Skörd ml', 'Microbiol']]

# %%
['Date', 'Temp_mean', 'Temp_max', 'Temp_min', 'Temp_std', 'pH_mean',
       'pH_max', 'pH_min', 'pH_std', 'Conductivity_mean', 'Conductivity_max',
       'Conductivity_min', 'Conductivity_std', 'CO2_mean', 'CO2_max',
       'CO2_min', 'CO2_std', 'dag', 'vecka', 'Ljus %', 'Ljus i watt',
       'Beräknad korrigerad kWh per dygn', 'Mätt ljuseffekt kWh per dag',
       'tot Ljus kWh', 'Ljussumma Sol LUX', 'CO2_x', 'temp Vmedel',
       'temp Vmax', 'temp Vmin', 'TEMP SUMMA_x', 'skörd (L pelletvolym)',
       'Diatom skörd', 'glidande medel diatom 3', 'glidande medel diatom 5',
```

```
        'Kontamination', 'Skörd ml', 'Microbiol', 'Förväntad Algica 15% Yeild',
        'CO2_y', 'Medeltemp 24h', 'TEMP SUMMA_y', 'LUX 1h', 'kWh per dag',
        'Ljussumma']

# %%
df[(df['Year'] == 2024) & (df['vecka'] == ...
    5)]['Kontamination'].mean(numeric_only=True)

# %%
weekly_aggregates = []

aggregate_columns = [
                'Temp_mean',
                'pH_mean',
                    'Conductivity_mean',
                    'CO2_mean',
                    'Medeltemp 24h',
                    'TEMP SUMMA_y',
                    'LUX 1h',
                    'kWh per dag',
                    'Ljussumma',
                    'Ljus %',
                    'Ljus i watt',
                    'Beräknad korrigerad kWh per dygn',
                    'Mätt ljuseffekt kWh per dag',
                    'tot Ljus kWh',
                    'Ljussumma Sol LUX',
                    'CO2_x',
                    'temp Vmedel',
                    #'temp Vmax', 'temp Vmin',
                    'TEMP SUMMA_x']

gustav_columns = ['CO2_x','Medeltemp 24h','TEMP SUMMA_x','LUX 1h','kWh per ...
    dag','Ljussumma']
bad_odlings_columns = ['Ljus i watt', 'Ljussumma Sol LUX', 'Mätt ljuseffekt kWh ...
    per dag', 'TEMP SUMMA_y', 'Beräknad korrigerad kWh per dygn', 'temp Vmedel', ...
    'tot Ljus kWh']
aggregate_columns = [col for col in aggregate_columns if col not in gustav_columns]
aggregate_columns = [col for col in aggregate_columns if col not in ...
    bad_odlings_columns]
target_columns = ['Diatom skörd',
                  #'skörd (L pelletvolym)',
                  'Kontamination',
                  #'Skörd ml',
                  'Microbiol']

#remove things with max and min from aggregate_columns
def read_aggregate(year, week):
    agg1 = df[(df['Year'] == year) & (df['vecka'] == ...
        week-3)][aggregate_columns].mean(numeric_only=True)
    agg2 = df[(df['Year'] == year) & (df['vecka'] == ...
        week-2)][aggregate_columns].mean(numeric_only=True)
    agg3 = df[(df['Year'] == year) & (df['vecka'] == ...
        week-1)][aggregate_columns].mean(numeric_only=True)
    # take mean of agg1, agg2, agg3 but keep in mind some can be nan
    agg = pd.concat([agg1, agg2, agg3], axis=1).mean(axis=1, skipna=True)
    week_minus_2 = df[(df['Year'] == year) & (df['vecka'] == ...
        week-3)][aggregate_columns].mean(numeric_only=True)
    week_minus_1 = df[(df['Year'] == year) & (df['vecka'] == ...
        week-2)][aggregate_columns].mean(numeric_only=True)
    week_mean = df[(df['Year'] == year) & (df['vecka'] == ...
        week-1)][aggregate_columns].mean(numeric_only=True)
    targets = df[(df['Year'] == year) & (df['vecka'] == week) & ...
        (df['Date'].dt.weekday == 0)][target_columns]
    # combine the three weeks into one row, rename columns with prefix w-2,w-1,w
    #week_df = pd.concat([week_minus_2, week_minus_1, week_mean, agg], axis=0)
    #week_df.index = [f'{col}_w-2' for col in aggregate_columns] + [f'{col}_w-1' ...
        for col in aggregate_columns] + [f'{col}_w' for col in aggregate_columns] ...
        + [f'{col}_agg' for col in aggregate_columns]
```

```
        week_df = pd.concat([agg], axis=0)
        week_df.index = [f'{col}_agg' for col in aggregate_columns]


        week_df['vecka'] = week
        week_df['Year'] = year
        week_df['Date'] = df[(df['Year'] == year) & (df['vecka'] == week) & ...
            (df['Date'].dt.weekday == 0)]['Date'].values[0]
        # convert date to numeric
        week_df['Date_numeric'] = pd.to_datetime(week_df['Date']).timestamp()
        # add to the week_df the target values
        # add columns with prefix target_
        for col in target_columns:
            week_df[col] = targets[col].values[0]
        return week_df

    for week in range(38, 52):
        weekly_aggregates.append(read_aggregate(2023, week))


    for week in range(4, 44):
        weekly_aggregates.append(read_aggregate(2024, week))

    # %%
    #get columns
    weekly_aggregates[0].index

    # %%
    weekly_aggregates[-1]

    # %%
    df_all = pd.DataFrame(weekly_aggregates)
    df_all = df_all.reindex(sorted(df_all.columns), axis=1)
    df_all.head()

    # %%
    # rename some columns
    df_all = df_all.rename(columns={'Temp_mean_agg': 'Temperatur',
                                'Microbiol': 'PAS',
                                'Date_numeric': 'Tid',
                                'pH_mean_agg': 'pH',
                                'Conductivity_mean_agg': 'Konduktivitet',
                                'CO2_mean_agg': 'CO2',
                                'Diatom skörd': 'Kiselalgsskörd',
                                'Ljus %_agg': 'Ljus %'})
    df_all.columns


    # %%
    import matplotlib.pyplot as plt
    # plot all variables against time in one seperate subplots columns, 'Volym ...
        systemtank A', 'pH - A', 'Antal aktiva banor - A','Tillfört CO2   - A', ...
        'Konduktivitet', 'Temp A', 'Skördsump_A'
    fig, axs = plt.subplots(4, 1, figsize=(10, 10))
    fig.suptitle('Treveckors genomsnitt bak, valda parametrar', fontsize=16)
    df_fig = df_all.copy()
    axs[0].plot(df_fig['Date'].values, df_fig['Temperatur'].values)
    axs[0].set_ylabel('Temperatur')
    axs[1].plot(df_fig['Date'].values, df_fig['pH'].values)
    axs[1].set_ylabel('pH')
    axs[2].plot(df_fig['Date'].values, df_fig['CO2'].values)
    axs[2].set_ylabel('CO$_2$')
    axs[3].plot(df_fig['Date'].values, df_fig['Konduktivitet'].values)
    axs[3].set_ylabel('Konduktivitet')
    #axs[4,1].plot(df_fig['Date'].values, df_fig['kWh per dag'].values)
    #axs[4,1].set_ylabel('kWh per dag')
    #axs[5,1].plot(df_fig['Date'].values, df_fig['Ljussumma'].values) # .interpolate()
    #axs[5,1].set_ylabel('Ljussumma')
    axs[0].get_xaxis().set_visible(False)
```

```
axs[1].get_xaxis().set_visible(False)
axs[2].get_xaxis().set_visible(False)
#axs[3,1].get_xaxis().set_visible(False)
#axs[4,1].get_xaxis().set_visible(False)
axs[3].tick_params(axis='x', rotation=45)

plt.tight_layout()
plt.savefig('finalfigs/tree_week_avg_data.png')
plt.show()

# %% [markdown]
# Korrelation Start

# %%
# show correlation with target columns
import seaborn as sns
import matplotlib.pyplot as plt
agg_col_names = ['Temperatur', 'pH', 'Konduktivitet', 'CO2', 'Ljus %', 'Tid']
target_columns = ['Kiselalgsskörd', 'Kontamination', 'PAS']


fig, ax = plt.subplots(figsize=(20, 20))
corr = df_all.corr()
def calculate_pvalues(df):
    dfcols = pd.DataFrame(columns=df.columns)
    pvalues = dfcols.transpose().join(dfcols, how='outer')
    for r in df.columns:
        for c in df.columns:
            try:
                tmp = df[df[r].notnull() & df[c].notnull()]
                pvalues.loc[r,c] = pearsonr(tmp[r], tmp[c])[1]
            except:
                continue
    return pvalues

corr_target =corr[target_columns].loc[agg_col_names]
pvalues = calculate_pvalues(df_all[agg_col_names + target_columns])

#for t_col in target_columns:
#    for agg_col in agg_col_names:
#        if pvalues[t_col][agg_col] > 0.05:
#            corr_target[t_col][agg_col] = np.nan




sns.heatmap(corr_target, vmin=-1, vmax=1, ...
    annot=True,linewidth=.5,annot_kws={"size": 24})
plt.title('Korrelation treveckors medelvärden', fontdict={'fontsize': 26}) # size ...
    of title
ax.set_yticklabels(ax.get_yticklabels(), fontsize=20)
ax.set_xticklabels(ax.get_xticklabels(), fontsize=20)
# size of title

#plt.title('Correlations of all data')
plt.savefig('finalfigs/correlations_3week_avg.png',bbox_inches='tight')

plt.show()

from scipy.stats import pearsonr
import pandas as pd

# print top 5
np.abs(corr_target['Kiselalgsskörd']).sort_values(ascending=False).head(20)

# %%
# korrelation aggregate med aggregate
plt.figure(figsize=(20, 20))
df_all.corr()
```

```
    corr_aggs = df_all.corr()[agg_col_names].loc[agg_col_names]
    sns.heatmap(corr_aggs, vmin=-1, vmax=1, ...
        annot=True,linewidth=.5,annot_kws={"size": 24})
    plt.title('Korrelation treveckors medelvärden', fontdict={'fontsize': 26}) # size ...
        of title
    ax.set_yticklabels(ax.get_yticklabels(), fontsize=30)
    ax.set_xticklabels(ax.get_xticklabels(), fontsize=10)
    # size of title

    #plt.title('Correlations of all data')
    plt.savefig('finalfigs/correlations_3week_avg_aggs.png',bbox_inches='tight')

    plt.show()

    # %%
    for c_target in target_columns:
        top_corrs_names = ...
            corr[c_target].loc[agg_col_names].abs().sort_values(ascending=False).head(10).index
        top_corrs = corr[c_target].loc[top_corrs_names].head(10)

        sns.heatmap(top_corrs.to_frame(), vmin=-1, vmax=1, ...
            annot=True,linewidth=.5,annot_kws={"size": 10})
        plt.title(f'Top 10 absoluta korrelation mot {c_target}', ...
            fontdict={'fontsize': 10}) # size of title
        # size of title

        #plt.title('Correlations of all data')
        plt.savefig(f'finalfigs/correlations_top_{c_target}.png',bbox_inches='tight')

        plt.show()

    # %% [markdown]
    # XGBOOST Start

    # %%
    import shap
    import xgboost as xgb
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import mean_squared_error, r2_score
    from sklearn.model_selection import KFold
    import matplotlib.pyplot as plt
    import seaborn as sns
    for t_col in target_columns:
        df_temp = df_all.copy()
        df_temp = df_temp.dropna(subset=[t_col])
        X = df_temp.drop(columns=['Date','vecka', 'Year'] + target_columns)


        y = df_temp[t_col]
        y = y/ max(y)

        print(X.shape)
        kf = KFold(n_splits=len(X))

        smse = []

        for train_index, test_index in kf.split(X):
            X_train, X_test = X.iloc[train_index], X.iloc[test_index]
            y_train, y_test = y.iloc[train_index], y.iloc[test_index]

            reg = xgb.XGBRegressor()
            reg.fit(X_train, y_train)
            y_pred = reg.predict(X_test)
            smse.append(np.sqrt(mean_squared_error(y_test, y_pred)))
        print(f'RMSE: {np.mean(smse):.2f}')

        reg = xgb.XGBRegressor()
        reg.fit(X, y)
        explainer = shap.Explainer(reg)
```

```
        shap_values = explainer(X)
        plt.title(f'SHAP värden XGBoost {t_col}')
        shap.plots.bar(shap_values, max_display=10, show=False)
        ax = plt.gca()
        for bar in ax.patches:
            bar.set_color('tab:blue')
        plt.savefig(f'finalfigs/shap_{t_col}.png',bbox_inches='tight')
        plt.show()

    # %%
    for t_col in target_columns:
        df_temp = df_all.copy()
        df_temp = df_temp.dropna(subset=[t_col])
        X = df_temp.drop(columns=['Date','vecka', 'Year'] + target_columns)

        y = df_temp[t_col]
        y = y/ max(y)

        print(X.shape)
        kf = KFold(n_splits=len(X))

        reg = xgb.XGBRegressor()
        reg.fit(X, y)
        explainer = shap.Explainer(reg)
        shap_values = explainer(X)
        plt.title(f'Individuella SHAP värden XGBoost {t_col}')
        shap.plots.beeswarm(shap_values, max_display=10, show=False)
        plt.savefig(f'finalfigs/shap_swarm_{t_col}.png',bbox_inches='tight')
        plt.show()

    # %% [markdown]
    # #Linear regression start

    # %%
    X.isna().sum()/X.shape[0]

    # %%
    X.shape

    # %%
    import shap
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import mean_squared_error, r2_score
    from sklearn.model_selection import KFold
    from sklearn.linear_model import LinearRegression
    import matplotlib.pyplot as plt
    import seaborn as sns
    for t_col in target_columns:
        df_temp = df_all.copy()
        df_temp = df_temp.dropna(subset=[t_col])
        X = df_temp.drop(columns=['Date','vecka', 'Year'] + target_columns)
        X = X.fillna(X.mean())
        y = df_temp[t_col]
        y = y/ max(y)

        fig, axs = plt.subplots(2, 3, figsize=(20, 14))
        # loop for each category
        i = 0
        for col in X.columns:
            reg = LinearRegression()
            X_train = (X[col].values.reshape(-1, 1) - min(X[col])) / ...
                (max(X[col])-min(X[col]))
            reg.fit(X_train, y.values)

            #plot the regression line
            axs[int(i/3),i%3].scatter(X_train, y, label=col)
            axs[int(i/3),i%3].plot(X_train, reg.predict(X_train), label=col + ' ...
                regression line')
            axs[int(i/3),i%3].set_xlabel(col)
```

```
            axs[int(i/3),i%3].set_ylabel(t_col)
            axs[int(i/3),i%3].set_title(f'{col}, coef: {reg.coef_[0]:.2f}')
            axs[int(i/3),i%3].legend()
            i+=1
        plt.suptitle(f'Linjär regression {t_col}',fontsize=20,x=0.5,y=0.92)
        plt.savefig(f'finalfigs/linear_vs_{t_col}.png',bbox_inches='tight')
        plt.show()

    # %%
    import os

    rename_from = ['finalfigs/correlations_3week_avg.png',
                   'finalfigs/correlations_3week_avg_aggs.png',
                   'finalfigs/correlations_top_Kiselalgsskörd.png',
                   'finalfigs/correlations_top_Kontamination.png',
                   'finalfigs/correlations_top_PAS.png',
                   'finalfigs/shap_Kiselalgsskörd.png',
                   'finalfigs/shap_Kontamination.png',
                   'finalfigs/shap_PAS.png',
                   'finalfigs/shap_swarm_Kiselalgsskörd.png',
                   'finalfigs/shap_swarm_Kontamination.png',
                   'finalfigs/shap_swarm_PAS.png',
                   'finalfigs/linear_vs_Kiselalgsskörd.png',
                   'finalfigs/linear_vs_Kontamination.png',
                   'finalfigs/linear_vs_PAS.png',
                   'finalfigs/odlingsresultat_skörd.png',
                   'finalfigs/odlingsresultat_params.png',]
    rename_to = ['finalfigs/6.2.png',
                 'finalfigs/appe_korrelations_data_v_data.png',
                 'finalfigs/appe_6.2.1.png',
                 'finalfigs/appe_6.2.2.png',
                 'finalfigs/appe_6.2.3.png',
                 'finalfigs/6.3.1_A.png',
                 'finalfigs/6.3.2_A.png',
                 'finalfigs/6.3.3_A.png',
                 'finalfigs/6.3.1_B.png',
                 'finalfigs/6.3.2_B.png',
                 'finalfigs/6.3.3_B.png',
                 'finalfigs/appe_6.2.1_2.png',
                 'finalfigs/appe_6.2.2_2.png',
                 'finalfigs/appe_6.2.3_2.png',
                 'finalfigs/5.2.3.png',
                 'finalfigs/appe_odlingsresultat_params.png']

    for from_file, to_file in zip(rename_from, rename_to):
        if os.path.exists(to_file) and os.path.exists(from_file):
            os.remove(to_file)

        if os.path.exists(from_file):
            os.rename(from_file, to_file)
            print(f'Renamed {from_file} to {to_file}')
        else:
            print(f'{from_file} does not exist')
```

B.5 XGBoost hyperparameteroptimering

Nedan följer pythonfilen som användes för att träna XGBoost-modell på olika dataset med hyperparameteroptimeringsmetoden slumpmässig sökning. Namn på filer och titlar på grafer är exempel och varierade beroende på dataset.

```
    import shap
    import numpy as np
    import pandas as pd
    import xgboost as xgb
    import statistics
    from scipy.stats import norm
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as MSE
import matplotlib.pyplot as plt


num_val = 4 #Hur många olika valideringsset den testas mot
branch_rounds = 15 #Hur många gånger den tar nya random hyperparametrar
branching_factor = 0.9 #Vad standardavvikelserna multipliceras med varje ...
    branching runda
rmse_model_val = []
for n in range(num_val):
    X = pd.read_csv(r'..\xdata_3vbak_skörd_co2_träning_norm.csv')
    y = pd.read_csv(r'..\ydata_3vbak_skörd_co2_träning_norm.csv')

    #Valideringsuppdelning, rörs inte förens sluttest
    X, X_val, y, y_val = train_test_split(X, y, test_size=0.15, random_state=n)

    # Hyperparameter standardvärden
    default_params = {
        "learning_rate": 0.3,
        "num_boost_round": 10,
        "max_depth": 6,
        "gamma": 0,
        "min_child_weight": 1,
        "subsample": 1.0,
        "reg_alpha": 0,
        "reg_lambda": 1
    }
    #Hur långt ifrån medel den letar
    std_devs = {
        "learning_rate": 0.1,
        "num_boost_round": 5,
        "max_depth": 2,
        "gamma": 1,
        "min_child_weight": 0.5,
        "subsample": 0.1,
        "colsample_bytree": 0.1,
        "reg_alpha": 0.2,
        "reg_lambda": 0.2
    }
    #min och maxvärden för varje hyperparameter
    bounds = {
        "learning_rate": (0.01, 1.0),
        "num_boost_round": (5, 500),
        "max_depth": (1, 15),
        "gamma": (0, 100),
        "min_child_weight": (1, 10),
        "subsample": (0.5, 1.0),
        "colsample_bytree": (0.5, 1.0),
        "reg_alpha": (0, 10),
        "reg_lambda": (0, 10)
    }


    results = []
    num_trials = 20 #Hur många olika random hyperparametervärden som testas varje ...
        branchrunda
    num_random_states = 5 #Hur många olika random states som testas för varje ...
        random hyperparametervärdeset
    rmse_prev = 1
    for k in range(branch_rounds):
        for i in range(num_trials):
            random_params = {
                key: np.clip(norm.rvs(loc=default_params[key], ...
                    scale=std_devs[key]), *bounds[key])
                for key in default_params
            } #genererar slumpmässiga hyperparametervärden från normalfördelning
            #Avrundar till int om hyperparametervärde måste vara heltal
```

```
            random_params["max_depth"] = int(round(random_params["max_depth"]))
            [language=python, numbers=none]
            random_params["booster"] = "dart" #bestämmer booster


            rmse_model = []
            for rand_state in range(num_random_states):
                X_train, X_test, y_train, y_test = train_test_split(X, y, ...
                    test_size=0.2, random_state=rand_state) #Ny train test split ...
                    med olika random states

                dtrain = xgb.DMatrix(X_train, label=y_train)
                dtest = xgb.DMatrix(X_test, label=y_test)

                #modellen tränas
                xgb_r = xgb.train(params=random_params, dtrain=dtrain, ...
                    evals=[(dtest, "Test")], ...
                    num_boost_round=random_params['num_boost_round'] )

                #modellen bedöms och rmse sparas
                pred = xgb_r.predict(dtest)
                mse = MSE(y_test, pred)
                rmse = np.sqrt(mse)
                rmse_model.append(rmse)

            rmse_avg = sum(rmse_model) / len(rmse_model) #Snitt på rmse tas för ...
                alla random states
            rmse_median = statistics.median(rmse_model) #median av rmse för alla ...
                random states av en viss hyperparameterinställning
            random_params["rmse_median"] = rmse_median
            random_params["rmse_avg"] = rmse_avg #snitt och median läggs till i ...
                resultat
            results.append(random_params) #slumpmässiga hyperparametrar läggs ...
                till i results


        results_df = pd.DataFrame(results)
        results_df = results_df.sort_values(by="rmse_median")  #resultat sorteras ...
            efter bäst median
        best_params = results_df.iloc[0].to_dict() #plockar ut den bästa raden i ...
            results


        if best_params['rmse_median'] ≤ rmse_prev:
            print(f'Best params: {best_params}')
            rmse_prev = best_params['rmse_median']
            del best_params['rmse_median']
            del best_params['rmse_avg']
            del best_params['booster'] #tar bort kolumner som inte är ...
                slumpmässiga hyperparametrar
            default_params = best_params #nya medelvärden för nästa rundas ...
                hyperparametrar är denna rundans bästa om den är bättre än förra

        std_devs = {k:v*branching_factor for k,v in std_devs.items()} #alla ...
            standardavvikelser multipliceras med branching factor

    #Efter flera rundor av detta tränas modellen om med bästa hyperparametrar, ...
        använder alla x utom valideringsdata

    dtrain = xgb.DMatrix(X, label=y)
    dval = xgb.DMatrix(X_val, label=y_val)
    best_params["booster"] = "dart" #bestämmer booster
    #best_params['tree_method'] = 'approx'
    xgb_best = xgb.train(params=best_params, dtrain=dtrain, evals=[(dval, ...
        "Validation")],num_boost_round=best_params['num_boost_round'] )

    #valideras på valideringsset
    pred = xgb_best.predict(dval)
    mse = MSE(y_val, pred)
```

```
        rmse = np.sqrt(mse)
        rmse_model_val.append(rmse)
        print(f"RMSE for trial {n + 1}: {rmse_model_val}") #RMSE på valideringsset


    importance_dict = xgb_best.get_score(importance_type='gain')
    print(importance_dict)

    #Visar importance dict för senaste bästa modellen
    for feat_name, importance_val in importance_dict.items():
        print(f"Feature {feat_name}: {importance_val}")
    final_rmse = np.mean(rmse_model_val)
    print(f"Final RMSE across all trials: {final_rmse}")

    X = pd.read_csv(r'..\xdata_3vbak_skörd_co2_träning_norm.csv')
    y = pd.read_csv(r'..\ydata_3vbak_skörd_co2_träning_norm.csv')
    dtrain = xgb.DMatrix(X, label=y)

    xgb_shap = xgb.train(params=best_params, ...
        dtrain=dtrain,num_boost_round=best_params['num_boost_round'] )
    #gör shap-plots
    explainer = shap.Explainer(xgb_shap)
    shap_values = explainer(X)

    plt.title('SHAP-värden XGBoost Kiselalgskörd')
    shap.plots.bar(shap_values, max_display=10, show=False)

    ax = plt.gca()

    for bar in ax.patches:
        bar.set_color('tab:blue') #ändrar färg på plot

    plt.show()

    plt.title('Individuella SHAP-värden XGBoost Kiselalgskörd')
    shap.plots.beeswarm(shap_values)
```