Norges miljø- og biovitenskapelige universitet

Masteroppgave 2022 30 stp
Handelshøyskolen

Test av godhet av kostnadsdrivere innen Activity-Based Costing.

Marius Ho Yung Agard Stenberg
Master i Økonomi og Administrasjon

Forord

Masteroppgaven understreker enden av en master i Økonomi og Administrasjon ved NMBU, med Business Analytics som spesialisering. Først vil jeg takke kjæreste Huynh Bao Ngan for all støtte det siste året. Deretter vil jeg takke Øystein Dahl for faglig støtte og behjelpelighet som veileder.

I tillegg vil jeg takke samarbeidende bedrift som har sørget for data og ytterligere veiledning, slik at prosjektet kunne gjennomføres.

Sammendrag

Hensikten med masteroppgaven er å optimalisere en kostnadsdriver innen Activity-based Costing ved hjelp av maskinlæring. Dette gjøres for å mulig utarbeide en generaliserbar metodikk som kan bidra til å bedre nøyaktigheten og relevansen til kostnadsdrivere. I tillegg undersøkes det om konkrete varer eller plukkområder påvirker plukktiden av et sett ordrelinjer når den aktuelle arbeidsaktiviteten er plukk av varer.

Den primære algoritmen som brukes er Support vector machines som blir optimalisert ved hjelp av hyperparametertuning og feature selection-metoder som regresjon, Pearsons R og variance threshold. Modellene sammenlignes også med Decision trees, K-nearest neighbors og Random forest ved bruk av cross validation. Avslutningsvis skal de beste versjonene fra alle disse algoritmene kombineres til en samlet modell ved hjelp av en metode som kalles stacking ensemble method.

Resultatene fra SVM-modellene viste at optimaliseringen førte til en minimal forbedring i prediksjonsevnen, men som totalt sett ikke var godt nok til å kunne generaliseres. I tillegg var det noen av algoritmene som hadde neglisjerbar, men bedre nøyaktighet enn SVM. Dette førte enten til en stacking-modell som var tilsvarende god som SVM-modellene eller en endelig stacking-modell som ikke fanget opp plukktiden godt nok. I tillegg tilsier resultatene at endring i plukktiden ikke forklares tilstrekkelig godt nok av enten varer eller plukkområder. Dette understreker at antall ordrelinjer som kostnadsdriver for plukkaktiviteten, består som en tilstrekkelig god kostnadsdriver.

Studien innehar noen implikasjoner i form av observasjoner på ca. 3 måneder som begrenser antall observasjoner tilgjengelig. En annen utfordring var et lavt antall relevante variabler som var tilgjengelig for å forklare plukktiden for et sett ordrelinjer. Sistnevnte implikasjon kan være et resultat av menneskelig feil under data-cleaning prosessen. For videre forskning som undersøker lignende fenomen, vil det være aktuelt å samle inn data over en lengre tidsperiode eller bruke simulert data.

Abstract

The purpose of this thesis is to optimize a cost driver in Activity-based costing by using machine learning. This is done to develop a generalizable method which can ultimately better the accuracy and relevance of cost drivers. In addition, i will further investigate whether specific goods or picking areas influences the picking time of order lines in a warehouse. In this instance, the work activity is picking of goods, whereas the cost driver is picking time of a set of order lines.

The primary algorithm is Support vector machines that's being optimized by hyperparameter tuning and various feature selection methods like regression, Pearson's R and variance threshold. The SVM-models are also being compared against Decision trees, K-nearest neighbors, and Random Forest by using cross validation. Finally, the best versions of these algorithms are being combined into a single model by using a method named stacking ensemble method.

The results from the SVM-models shows that the optimization led to a minimal increase in predictive ability, but overall fell short of showing good predictive ability. Furthermore, some of the other algorithms were slightly more accurate than SVM, but these positive results were negligible. This either led to a stacking-model which was as good as the SVM-models or a final stacking-model which didn't catch the complexity of the dependent variable. Also, the results states that differences in picking time are not explained fully by either specific goods or picking areas. In total, this implies that quantity of order lines remains a good cost driver for picking activity in a warehouse.

First implication is the limited observations over 3 months. Second implication is the low number of features/variables available. Although the last implication might be a result of human error in the data cleaning process. For future research I would recommend either to gather data over a larger time frame or use simulated data.

Innhold

Tabeller

Figurer

Forklaring av forkortelser

ABC: Activity-based costing

AIC: Akaike information criterion

ANN: Artificial neural network

CV: Cross validation

DT: Decision trees

FS: Feature selection

KNN: K-nearest neighbors

MCC: Matthews korrelasjonskoeffisient

MSE: Mean squared error

RBF: Radial basis function

RF: Random Forest

SVM: Support Vector Machines

1. Introduksjon

1.1 Bakgrunn

Aktivitetsbasert kostnadskalkulasjon er ifølge Bjørnenak (1993) en metode som blant annet undersøker sammenhengen mellom indirekte ressurser, aktiviteter og kostnadsdrivere, for å knytte kostnadene opp mot kunder, produkter eller produktgrupper. En aktivitet kan være en konkret arbeidsoppgave som plukk, registrering eller bestilling av varer. Samtidig vil hver arbeidsoppgave kreve forskjellige ressurser. Eksempelvis kan et lager med forskjellige plukkområder ha behov for ulikt antall personer, utstyr og maskiner for å utføre jobben. En kostnadsdriver er det som skaper kostnadene til en aktivitet. Eksempelvis vil aktiviteten plukk av varer, skape en kostnad basert på antall ordrelinjer som er bestilt.

Maskinlæring har eksistert i lang tid, men har de siste årene blitt mer brukt for å løse statistiske problemstillinger ved hjelp av algoritmer. Økt bruk de senere årene kan skyldes tilgang på større mengder data og bedre effektivitet innen prosessering. Hovedhensikten til maskinlæring er blant annet predikering og klassifisering, hvor algoritmene har behov for store mengder data. Valg av algoritme vil avhenge av hva den konkrete algoritmen blir brukt til, hva slags data en ønsker å undersøke, størrelsen på datasettet og hva som ønskes predikert eller klassifisert. Klassifikasjonsalgoritmer er algoritmer hvor modellen grupperer de uavhengige variablene, basert på input i observasjonene som er gitt i datasettet. (Nichols et al., 2018).

Hittil har det blitt undersøkt om ulike maskinlæringsalgoritmer kan bli implementert for å optimalisere kostnadsdrivere i en ABC-analyse (Kim & Han, 2003: Babad & Balachandran, 1993). En bemerkning til flere av artiklene, er at flere studier er blitt gjennomført med simulert data og ikke en praktisk case med data fra en reell bedrift. Det vil derfor være interessant å introdusere ABC-fagområdet til en annen algoritme (Support vector machines-SVM) og gjøre dette på premisset til et operasjonelt datasett. Hvorvidt SVM er generaliserbar for optimalisering av kostnadsdrivere og om det kan integreres med ABC er bakgrunnen for denne oppgaven. I tillegg er det aktuelt å kartlegge hvilke uavhengige variabler som påvirker tid på å utføre plukk av et sett ordrelinjer som kostnadsdriver og hvorvidt andre algoritmer presterer bedre enn SVM.

1.2 Hensikt, problemstilling og hypoteser

Formålet er å utforske hvordan maskinlæring kan skape merverdi til ABC gjennom algoritmen SVM for optimalisering av kostnadsdrivere. Ved å bruke maskinlæring for å finne hva som påvirker plukktiden for et sett ordrelinjer, vil det bli avdekket hvorvidt påføringen av algoritmen er generaliserbar for andre bedrifter og hvorvidt dette er interessante funn for ABC eller ikke, i henhold til optimalisering av kostnadsdrivere.

I tillegg vil jeg belyse forskjellige maskinlæringsalgoritmer sine bidrag til optimalisering av kostnadsdrivere innen ABC fra studier som stammer tilbake fra 90-tallet. Fellesnevneren for noen av disse studiene er bruken av simulert data. Av den grunn er det hensiktsmessig å analysere datasett fra bedrifter som kan ha nytte av enderesultatet.

Basert på formål og problemstilling har det blitt utarbeidet følgende hypoteser:

> **Hypotese 1**: Support vector machines er ikke en egnet algoritme til å optimalisere kostnadsdrivere.
>
> **Hypotese 2**: Andre algoritmer presterer bedre enn SVM i henhold til å predikere tiden ved å plukke et sett ordrelinjer.
>
> **Hypotese 3**: SVM/Stacking-modellene kan predikere forskjell i plukktiden for forskjellige varer.
>
> **Hypotese 4**: SVM/Stacking-modellene kan predikere forskjell i plukktiden for forskjellige plukkområder.

1.3 Masteroppgavens struktur

Innledningsvis vil introduksjonen gjøre rede for bakgrunnen for oppgaven, hensikt, problemstilling og hypoteser. Deretter vil teoridelen vise til relevant forskning og teori. Først vil ABC bli introdusert, i tillegg til optimalisering av kostnadsdrivere ved hjelp av forskjellige algoritmer og matematiske modeller som en avrunding av teorien. Etter dette følger metodekapittelet som tydeliggjør fremgangsmetoden jeg har brukt for å svare på problemstillingen og hypotesene. Deretter følger datakapittelet hvor jeg gjør rede for hvilke variabler som brukes i analysen. Data etterfølges av resultater fra analysen, hvor det deretter blir gjennomført en diskusjon som fører til konklusjoner som bygger på hypotesene og problemstillingen.

2. Teori

Teoridelen gjør rede for relevant teori i henhold til kostnadsdriveroptimalisering i ABC. Først introduseres grunnleggende prinsipper i ABC, i tillegg til noe innledende metodikk. Videre vil konkrete kostnadsdriverkriterier og tilnærminger bli redegjort for. Avslutningsvis skal det presenteres flere eksempler på kostnadsdriveroptimalisering ved bruk av forskjellige maskinlæringsalgoritmer.

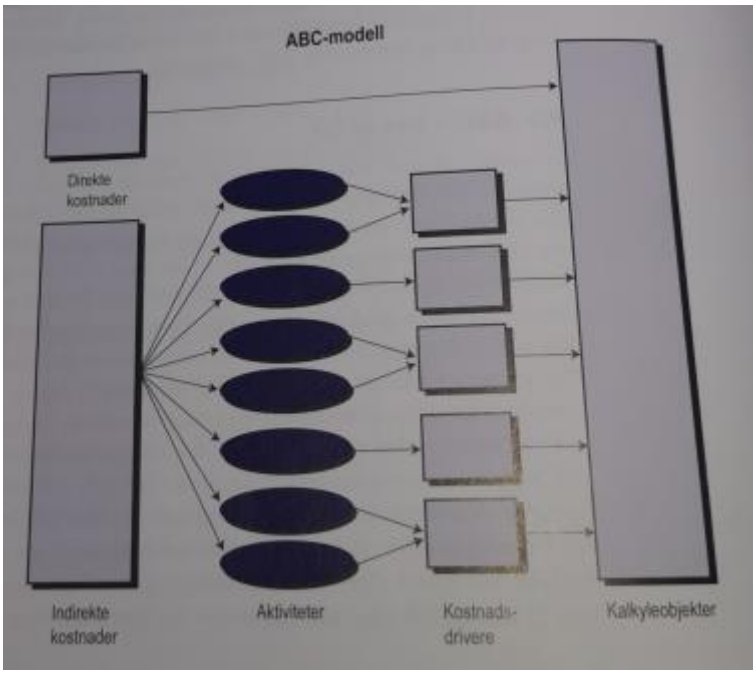
2.1 Grunnleggende metodikk og prinsipper ved Activity-based Costing (ABC)

Innledningsvis vil jeg introdusere hovedkonsepter innen ABC og noen grunnleggende prinsipper, men ikke gå i dybden av selve utførelsen av en ABC-analyse.

Cooper & Kaplan (1988) introduserte ABC på slutten av 80-tallet, hvor de innførte en ny metode som fordelte kostnader til enkelte produkter/produktgrupper eller kunder for de fleste funksjonene i en bedrift. Videre hevder de at ABC er et strategisk verktøy som skal gi en bedre oversikt over kostnadsallokeringen, som deretter blir brukt for å ta veloverveide strategiske beslutninger. (Cooper & Kaplan, 1988) Med andre ord kan en ABC-analyse gi nærmere innsikt i detaljerte kostnader tilknyttet en kunde eller et produkt.

Metodikk og prinsipper ved ABC

ABC sin metodikk og grunnmodell er illustrert i figuren nedenfor (Hoff et al., 2017). Modellen viser at kalkyleobjektene i form av produkter eller kunder tillegges både direkte og indirekte kostnader. Den detaljerte ressursfordelingen oppstår når en legger til indirekte kostnader til arbeidsaktiviteter. Med andre ord hva slags kostnader som en må regne med for å gjennomføre arbeidsaktiviteten. I neste steg tillegges aktivitetene en kostnadsdriver som utgjør de indirekte kostnadene per enhet. Eksempelvis kan en kostnadsdriver være antall ordrelinjer, hvor aktiviteten kan være plukk av varer. Da vil kostnadsdriveren være tilknyttet per ordrelinje som blir plukket, selv om dette kan innebære en forskjell i antall varer som blir plukket per ordrelinje. Siste steg innebærer å legge til antall enheter av kostnadsdriverne til tilsvarende antall av forbruk for kalkyleobjekter (Hoff et al., 2017).



Figur 1: Grunnmodell i ABC (Hoff et al., 2017, 220)

Bjørndahl et al. (2003) nevner deres oppfatning av de grunnleggende prinsippene ved ABC. Det første er at ABC viser hvordan ressursene blir brukt gjennom aktiviteter, hvor tidligere kalkyler i større grad har fordelt ressursene på avdelinger. Det andre prinsippet tilsier at økt kompleksitet vil føre til økte kostnader. Neste prinsipp tydeliggjør kostnadsdriverne og hvordan denne variabelen belaster arbeidsaktivitetene. Siste prinsipp går ut på kapasitetsutnyttelse, hvor Bjørndahl et al. (2003) mener at kostnaden ved ubrukt kapasitet ikke skal tillegges kalkyleobjektene.

Basert på Cooper & Kaplan (1988), fremhever Bjørnenak (1993) viktigheten av aktiviteter og innsatsfaktorer. En aktivitet eller hjelpeaktivitet er det arbeiderne gjør for å bli ferdig med en konkret arbeidsoppgave. Bjørnenak (1993) beskriver også at aktivitetene krever innsatsfaktorer. Dette kan være ansatte, utstyr og andre ting som trengs for å gjennomføre en aktivitet. Eksempel på dette kan være å registrere varer på et varemottak, hvor enkelte spesialvarer kan ha en annen fremgangsmåte og protokoll når det kommer til registrering, kvalitetssjekk og videre lagring.

Fordeler og ulemper ved ABC

Hoff et al. (2017) nevner både fordeler og ulemper ved å bruke ABC. Første fordel tilsier at fordelingsnøkler for de indirekte kostnadene vil gi bedre innsikt i kalkyleobjektenes ressursbruk. I dette tilfellet vil fordelingsnøkler være omtalt som kostnadsdrivere, hvor fordelingsgrunnlaget kan tolkes som hvordan de indirekte kostnadene er fordelt på de ulike aktivitetene. Andre fordel er informasjon om hva aktiviteter koster og hvorvidt dette fører til økt lønnsomhet eller ikke for kalkyleobjektet. Med denne informasjonen kan bedriften avgjøre om de skal la aktiviteten være som den er, forandres, eller avvikles. Samtidig kan det være uaktuelt å kutte ut enkelte aktiviteter som er essensielle og er noe som må vurderes etter skjønn. I tillegg vil belysning av kapasitetsutnyttelsen og de følgende kostnadene til ledig kapasitet gi et bedre beslutningsgrunnlag. På den andre siden er det en ulempe hvis det koster mye å innhente og analysere kostnadsdriverne. Spesielt hvis det skal produseres detaljerte kostnadsdrivere som det tar lang tid å utvikle.

2.2 Kostnadsdrivere i ABC

En kostnadsdriver kan defineres på flere måter. For det første kan en kostnadsdriver være variabelen som forklarer endring i kostnadene for arbeidsaktiviteter (Bjørndahl et al, 2003). Eksempel på konkret kostnadsdriver hos et lager kan være antall varer som skal registreres i systemet ved innleggelse av varer på varemottak som arbeidsaktivitet. En annen definisjon er at kostnadsdrivere er enheter som er kausalt tilknyttet ressurskostnader som brukes til aktiviteter som ytterligere krever ressurser. Med andre ord er en kostnadsdriver det som endrer aktivitetskostnaden eller hvor mye ressurser som involveres for å gjennomføre en aktivitet (Cokins & Capusneanu, 2010).

Ifølge Cokins & Capusneanu (2010) er det flere kriterier å ta hensyn til ved valg av kostnadsdrivere, hvor de skiller mellom valgfrie og nødvendige kriterier. Det første valgfrie kriteriet innebærer hvor lett en kan finne kostnadsdriverne og hvor forståelige de er. Det bør være et tydelig forhold mellom aktiviteter og hvor mye indirekte kostnader som forbrukes. Videre understreker de viktigheten av årsakssammenheng mellom indirekte kostnader og kostnadsdrivere, da de mener at dette gir en nøyaktig fremstilling av kostnadene til aktivitetene. Kausaliteten er samtidig et omstridt krav, hvor Dahl et al. (2021) hevder at kostnadsdriveren ikke må drive kostnaden og heller kan fungere som en allokeringsnøkkel.

De nødvendige kriteriene for valg av kostnadsdrivere innebærer at kostnadsdriverne skal representere aktivitetskostnadene tilstrekkelig bra nok i forhold til hvert unike kostnadsobjekt. Kostnadsobjekt kan være konkrete kunder eller produkter. Eksempelvis kan det være stor forskjell på aktiviteter og påfølgende kostnadsdrivere mellom dyre håndlagde skinnsko og masseproduserte billige joggesko. Det andre nødvendige kostnadsdriverkriteriet er riktig seleksjon av antall kostnadsdrivere. Hvis det er for få eller for mange kostnadsdrivere vil beregningene for noen kostnadsobjekter være feilaktige og gi feil beslutningsgrunnlag. Siste kriteriet påpeker at utarbeidelsen av kostnadsdriverne, med grunnlag av de andre kriteriene er helt avhengig av hva slags beslutninger som skal tas (Cokins & Capusneanu, 2010). For de bedriftene som bruker ABC-analysen aktivt som et strategisk styringsverktøy, vil det være mer hensiktsmessig å være mer nøyaktig ved utarbeidelse av kostnadsdriverne enn for de bedriftene som ikke gjør det.

2.3 Kostnadsdriveroptimalisering i ABC

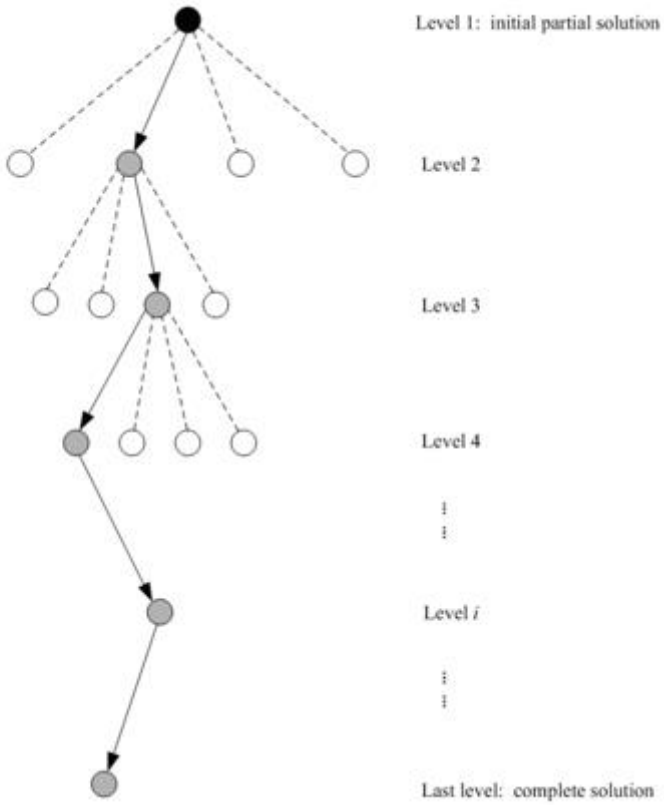
Kostnadsdriveroptimalisering baserer seg på å bruke algoritmer for å finne de best egnede kostnadsdriverne. I tillegg handler det om å finne antallet kostnadsdrivere som er best mulig. Med andre ord, gjøres dette for å vise hvilke kostnadsdrivere som best representerer kostnadene til arbeidsaktivitetene som er relatert til en kunde eller et produkt. Følgende delkapitler viser hvordan forskjellige maskinlæringsalgoritmer brukes for å optimalisere kostnadsdrivere i ABC.

Kostnadsdriveroptimalisering med composite greedy algorithm

Babad & Balachandran (1993) optimaliserte valg av kostnadsdrivere ved bruk av en composite greedy algorithm. De gjør det klart at det er viktig for et ABC-system å finne hvilke kostnadsdrivere som fanger opp aktivitetene med de mest signifikante kostnadsdriverne, og fant også at en kan slå sammen flere aktiviteter og relatere de til en samlet kostnadsdriver i tilfeller med korrelasjon.

En composite greedy algorithm er en algoritme som alltid velger den beste lokale løsningen. Totalt sett vil det endelige resultatet utgjøre den beste som algoritmen klarer å oppdrive, men ikke nødvendigvis være det beste hvis fremgangsmetoden var litt annerledes (GeeksForGeeks,

2022). Forklart annerledes vil algoritmen steg for steg sette en verdi for evalueringskriteria som avgjør hvilken løsning som skal bygges videre på (Chen, 2008). På modellen nedenfor kan en se at de grå prikkene på hvert nivå representerer den beste lokale løsningen på hvert steg. Disse er de beste lokale løsningene som algoritmen identifiserer på hvert steg, men tar ikke i betraktning i de totale forholdene i datasettet. Et praktisk og illustrativt eksempel kan være om et fotballag ønsker en god keeper og dermed kjøper en keeper i verdensklasse istedenfor å utvikle sin egen spillerbase eller kjøpe en keeper som er tilstrekkelig god nok med bra utviklingspotensialet. Da har fotballaget tatt den beste lokale løsningen og løst den nærmeste problemstillingen, men dette kan virke mot sin hensikt dersom de også trenger flere spesialiserte spillerstillinger som de nå ikke har midler til å løse på samme måte. Med andre ord har de sannsynligvis ikke lenger budsjett til å kjøpe en midtbane eller spiss i verdensklasse og må nøye seg med dårligere spillere i disse posisjonene. Den situasjonen vil ikke være gunstig totalt sett.



Figur 2: Representasjon av greedy algorithm. (Chen, 2008)

Ved å stegvis vektlegge evalueringskriterier for kostnadsdriverne med en composite greedy algorithm, klarte Babad & Balachandran (1993) å kombinere kostnadsdrivere med tilnærmet perfekt korrelasjon. Slik opprettholdt de fortsatt nøyaktigheten til kostnadsdriverne, selv om de totalt sett endte opp med færre kostnadsdrivere. Algoritmen tok også for seg de viktigste aktivitetene først, hvor de gradvis gikk over til å kombinere de aktivitetene som ikke var like signifikante (Babad & Balachandran, 1993).

Babad & Balachandran, 1993) baserte blant annet modellen sin på det faktum at bedriften måtte veie opp fordelene ved nøyaktige kostnadsdrivere, i motsetning til hva det ville koste å innhente og lagre denne dataen (Babad & Balachandran, 1993). Grunnet sistnevnte faktor, vil modellen sannsynligvis være litt utdatert i 2022, da datainnhenting, lagring og prosessering er blitt mer kostnadseffektivt og tilgjengelig de siste 30 årene.

Kostnadsdriveroptimalisering med genetisk algoritme og nevrale nettverk (ANN)

Kim & Han (2003) introduserte en fremgangsmetode hvor de brukte en genetisk algoritme og nevrale nettverk for å identifisere kostnadsdrivere, samt sammenheng mellom kostnader og aktiviteter som ikke nødvendigvis hadde en lineær korrelasjon. Forskjeller på nevrale nettverk og dens funksjoner kommer ikke til å beskrives i detalj, da dette går utover denne masteroppgavens omfang.

Innledningsvis benyttet Kim & Han (2003) seg av en genetisk algoritme for å finne best mulige kostnadsdrivere. En genetisk algoritme er basert på darwinisme, der de sterkeste overlever og reproduserer seg (Mallawaarachchi, 2017). Et annet praktisk eksempel på en genetisk algoritme kan være opplæring av en sjakkrobot sin kunstige intelligens. Dersom en hadde utviklet kunstig intelligens for en maskin som kun spilte sjakk og lot den spille mot seg selv i 10 forskjellige versjoner, så kunne en genetisk algoritme valgt ut de versjonene som var best og reprodusert en ny generasjon basert på egenskapene til sjakkrobotene som var best i sjakk. Dette kan fortsette videre i generasjoner, hvor versjonene blir bedre og bedre jo flere ganger de formerer seg.

Kort forklart er nevrale nettverk basert på menneskehjernens nevroner som interagerer med hverandre og tar avgjørelser basert på dette i tillegg til sanseinntrykk. Nevrale nettverk kan blant annet bli brukt til å gjenkjenne mønstre, prediksjon og optimalisering (Malekian, 2021). Et praktisk eksempel på et nevralt nettverk kan være utvikling av kunstig intelligens for en førerløs bil hvor den skal kjøre en bil rundt en oval kjørebane. Her kan nevronene representere økt hastighet, redusert hastighet, sving til venstre, syn framover og syn til siden. Det eneste den får beskjed om som input er å ikke kjøre utenfor banen, men komme til målstreken så fort som mulig. Alle de mulige handlingene kommuniserer med hverandre og vil få ulike utslag etter hvor bilen befinner seg. Dersom bilen nærmer seg en sving, vil den sannsynligvis sakke farten litt og samtidig begynne å svinge for å unngå å kjøre ut av banen. Samtidig vil den oppdage at det er mulig å øke farten og gradvis slutte å svinge når den er på vei ut av en sving. Sannsynligvis lærer den førerløse bilen seg dette på egenhånd ved å prøve og feile et par ganger.

Videre kombinerte Kim & Han (2003) den genetiske algoritmen med nevrale nettverk (Artificial neural network – ANN) og lagde en hybridmodell som fungerte bedre enn et standardisert nevralt nettverk. Dette var tydelig da hybridmodellen bestod av færre kostnadsdrivere, men høyere nøyaktighet, hvor et standard nevralt nettverk utgjorde flere kostnadsdrivere med lavere nøyaktighet ved allokering av indirekte kostnader (Kim & Han, 2003). Forskerne mener at en miks av genetisk algoritme og ANN er hensiktsmessig å benytte seg av hvis datasettet er stort og komplisert.

Ulempen med denne studien er at de brukte simulerte data og at de dermed ikke vet om funnene er generaliserbare til reelle datasett (Kim & Han, 2003). Utenom denne implikasjonen, oppfordret de videre forskning til å utforske optimalisering gjennom support vector machines.

Delavsnittene ovenfor har spesifisert flere metoder på kostnadsdriveroptimalisering ved hjelp av maskinlæring. Som nevnt tidligere, er det muligens et problem at analysene er utført på simulerte data og ikke et virkelig datasett. På en annen side er det hensiktsmessig å bruke denne metoden, dersom det kan utarbeides en generaliserbar modell som kan brukes av andre.

Samtidig skal analysen gjennomføres på et datasett som representerer virkelig arbeidsaktivitetsdata, hvor påfølgende kapittel forklarer metodikken.

3. Metode

Metodekapittelet vil først forklare datainnsamlingen, begrensninger og deretter hva som utgjør Support vector machines, hvor det også gjøres rede for hvordan modellen blir målt og vurdert. Metode fortsetter med forskjellige optimaliseringsteknikker av algoritmen som også innebærer cross validation og stacked ensemble learning. Hensikten er å finne hvilke avhengige variabler som påvirker plukktiden på et sett ordrelinjer og hvorvidt modellen som utarbeides er generaliserbar.

3.1 Datainnsamling

Innhenting av data tok form gjennom møter med samarbeidende bedrift, som hadde blitt kontaktet gjennom veileders nettverk. Kvantitative arbeidsprosessdata og ytterligere metadata om forsendelser og vareinformasjon ble overført fra samarbeidende bedrift i form av 18 Excel-dokumenter.

De primære datasettene bestod av arbeidsaktiviteter på et utvalg av to tilfeldige lagere som bedriften disponerte. Arbeidsaktivitetene er tradisjonelle logistikkaktiviteter som innebærer alt fra mottak av varer og helt til varen er pakket og sendt videre til kunden. I tillegg ble det mottatt informasjon om produkter, varegrupper og ordreforsendelser over en tidsperiode på 3 måneder.

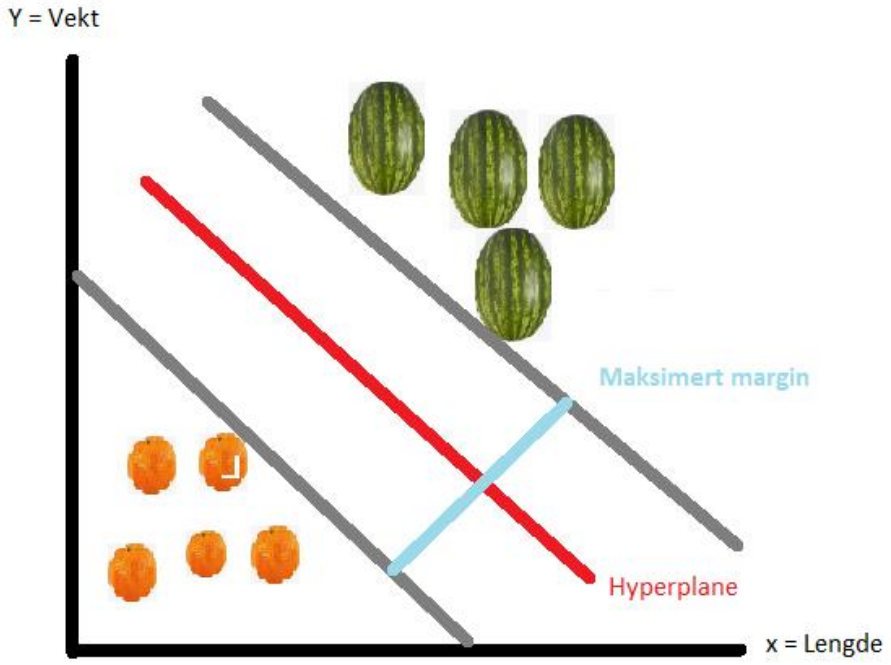
3.2 Support vector machines (SVM)

Support vector machines (SVM) er en algoritme innen maskinlæring som brukes for å predikere og klassifisere. De følgende delkapitlene viser oppbygningen av algoritmen og noen komplementære maskinlæringsprinsipper. Denne studien kommer ikke til å gå i dybden av matematikken bak algoritmen.

Hyperplane, maximal margin og soft margin.

Hyperplane i SVM er det samme som en skillelinje som deler klassifiseringene i to eller flere forskjellige klasser. SVM kan anses som en type klyngeanalyse hvor resultatet skiller grupperinger av observasjoner fra hverandre (Pupale, 2018). Nedenfor har jeg illustrert

hvordan dette kan se ut i praksis. For å lettere forklare konseptet har jeg brukt klementiner og vannmeloner som et utgangspunkt, der x-aksen tilsvarer lengden på frukten og y-aksen representerer vekten. Da frukten kun blir klassifisert basert på to dimensjoner, kan vi anse datasettet som 2-dimensjonelt.



Figur 3: SVM: Hyperplane og maksimert margin

For å danne skillelinjen så vil algoritmen finne de nærmeste punktene fra begge klassene og sette opp en optimal skillelinje midt imellom for å separere klassifiseringene. Avstanden mellom de to klassifiseringene i dette tilfellet tilsvare begrepet maksimert margin.

Soft margin er funksjonen som lar feilklassifiseringer være en del av modellen. Eksempelvis ville en rekordstor appelsin i modellen ovenfor ført til at den røde hyperplane-linjen beveget seg mer til høyre. Den rekordstore appelsinen representerer en ekstremverdi som bør ignoreres for at modellen skal fange opp mindre støy. Parameteren som justeres innen SVM er C. Dersom denne verdien tilsvarer 0, vil modellen fange opp alle ekstremverdier til sine respektive klasser. Hvis verdien er høyere, vil C la flere feilklassifiseringer gå gjennom, ifølge Brownlee (2016, b).

Feilklassifiseringer som skyldes ekstremverdier (outliers), må tas i betraktning for å plassere skillelinjen best mulig. For å bygge videre på illustrasjonen ovenfor kan det hende at en litt mindre vannmelon ble klassifisert som en klementin dersom den ikke var så stor eller veide like mye som de andre vannmelonene. Den lille vannmelonen ville da befunnet seg på venstre side av den røde skillelinjen. For å unngå at denne og andre små vannmeloner skal bli klassifisert som klementiner, kan vi tillate modellen å gjøre noen feil og anse de små vannmelonene som outliers.

Parametertuning: C og Gamma

På en annen side kan vi selv justere på SVM sin C-parameter som tilsier at en positiv perfekt verdi på +1 ikke tillater klassifiseringsfeil av treningssettet til maskinlæringsmodellen. En mulig konsekvens av en altfor høy C-verdi kan være overfitting. Ifølge IBM (2021) defineres overfitting som et konsept som skjer når en statistisk modell har like trenings- og testsett, slik at algoritmen ikke klarer å bruke datasettet for å predikere og klassifisere data. En positiv perfekt C-parameter er med andre ord ikke ønskelig, hvor det er hensiktsmessig å prøve ut forskjellige verdier i løpet av analyseforløpet.

Videre kan en også justere gammaverdien fra 0 til 1. Gamma med høy verdi vil tilsi at observasjoner som er nærme en klassifisering vil være sterkt påvirket. På en annen side vil lav gamma være lite påvirket, selv om observasjonen befinner seg nærme en klassifiseringsgruppe (Pedregosa et al., 2011).

Kernels

Kernels er en viktig del av hvordan SVM-algoritmen fungerer i praksis. For å danne grunnlaget for en optimal skillelinje, baserer SVM plasseringen av observasjonene på matematiske prinsipper. På illustrasjonen ovenfor ville det eksempelvis vært aktuelt å bruke en lineær kernel, da det kun var to variabler som ble målt for å klassifisere frukten. Min studie vil ha behov for en radial kernel. Årsaken er at det skal undersøkes mange variabler som eksempelvis kan påvirke hvor lang tid det tar å plukke ferdig en ordrelinje. Fordelen er at en kan legge inn så mange variabler som mulig i en radial kernel. Noe som ikke fungerer for

hverken den lineære- eller polynome kernel som ikke kan behandle så mange dimensjoner av gangen. Igjen vil jeg påpeke at gjennomgang av matematikken er utenfor denne studiens omfang, hvor de to andre kernels i SVM heller ikke vil bli utdypt.

OneHotEncoding

For at de fleste algoritmer skal forstå kategoriske data, må de gjøres om til tall ved hjelp av OneHotEncoding. Eksempelvis vil noen algoritmer ikke klare å kalkulere forskjellige varegrupper hvis de er satt opp som «kontorrekvisita» eller «hygieneprodukter», da disse gruppene ikke innehar noen numerisk verdi. Ved å gjennomføre OneHotEncoding gjør en om disse varegruppene til dummyvariabler i form av enten 0 eller 1 ifølge Brownlee (2017). En videreføring av eksempelet ovenfor kan vises i følgende tabell, hvor verdien tilsvarer 1 dersom varen er kontorrekvisita og 0 dersom den ikke er det.

Tabell 1: Visuell fremstilling av dummyvariabler

Kontorrekvisita	**Hygieneprodukter**
1	0
0	1

Accuracy og F1

For å måle hvor god en modell er, er det hensiktsmessig å måle accuracy og F1. Accuracy er omtalt som nøyaktighet og viser prosentandelen for hvor mange enheter som er riktig klassifisert (Huilgol, 2019). Eksempelvis om en modell skal klassifisere om en pasient har korona og er 70% nøyaktig, vil det si at modellen i 7 av 10 tilfeller klassifiserer riktig. På en annen side vil det være 30% som er klassifisert feil, noe som ikke er ønskelig, hvor det er nødvendig å introdusere F1 for å måle feilklassifiseringer i modellen. F1 tar hensyn til hvor nøyaktig en modell er, på bakgrunn av presisjon og recall (Huilgol, 2019). Presisjon justerer nøyaktigheten etter antall positive feildiagnostiseringer av korona. Recall måler nøyaktigheten, justert mot antall predikerte feildiagnostiseringer av korona. Med andre ord de som ikke hadde korona, men som ble diagnostisert ved en feiltakelse. Kombinert tilsvarer presisjon og recall, F1 som tar mer hensyn til feilklassifiseringer enn nøyaktighet alene (Huilgol, 2019).

SVM – Fordeler og ulemper

Fordelene og ulempene ved SVM er tydeligE i scikit-learn dokumentasjonen (Pedregosa et al., 2011). Fordelene med SVM er at algoritmen klarer å behandle mange dimensjoner av gangen og at modellen kan trenes, selv med mindre datasett. Sistnevnte er en stor fordel innen bransjer hvor data ikke blir lagret over lenger tid, hvor en har mindre data å jobbe med. Dette er også motsatt fra mange andre maskinlæringsalgoritmer som på den andre siden er avhengige av store datasett for å kunne predikere og klassifisere noe uten å være preget av overfitting.

Ulempen er at analytikeren selv må velge riktig kernel og C-parameter for å unngå overfitting. Algoritmen kan derfor by på utfordringer ved menneskelig feil.

3.3 Optimalisering innen maskinlæring

Innen maskinlæring finnes det flere metoder å optimalisere algoritmer på. Feature selection, videre omtalt som FS, er en blanding av ulike metoder som reduserer antall variabler for å redusere støy i datasettet og øke nøyaktigheten av analysen. I tillegg har FS som oppgave å plukke ut de viktigste variablene som forklarer et fenomen (Remeseiro & Bolon-Canedo, 2019).

Pearsons korrelasjonskoeffisient

Denne testen blir heretter omtalt som Pearsons R og blir brukt for å finne kovariasjon og korrelasjon mellom de resterende numeriske variablene. Pearsons R undersøker om noen av de uavhengige variablene assosieres med hverandre og måler det samme. Dersom det finnes høy kovariasjon som er tilnærmet +1 mellom to uavhengige variabler som forklarer tilnærmet lik varians i datasettet, bør en variabel ekskluderes for å optimalisere modellen. Den ekstra variabelen med tilsvarende trend vil kun være støy for utarbeidelsen av algoritmen (statisticssolutions, u.å.)

Variance threshold:

Varianche threshold er en metode innen maskinlæring for optimalisering av modeller. Hensikten er å fjerne alle variabler som ikke forklarer en satt grense av variansen. Vanligvis er denne grensen lik null, hvor testen fjerner alle variabler som ikke forklarer noe av variansen i datasettet (Pedregosa et al., 2011). Chandrashekar & Sahin (2014) mener variance threshold er en filtermetode som fjerner alle irrelevante variabler som ikke når opp til sperregrensen.

Cross validation

Cross validation er videre omtalt som CV og er en metode som brukes for å resample data og øke den prediktive nøyaktigheten til modellene den sammenligner (Berrar, 2019). Samtidig vil CV motarbeide overfitting. Et eksempel på overfitting vil være om en modell har 100% prediksjonsnøyaktighet hvor de uavhengige og den avhengige variabelen ikke har stor kovariasjon, men hvor modellen er blitt tilpasset slik at den ignorerer dette. I så fall vil modellen prestere bra med det konkrete datasettet, men vil være vanskelig å generalisere for helt ny data (Berrar, 2019). Berrar (2019) hevder samtidig at underfitting er et problem, hvor modellen ikke klarer å forstå sammenhengen mellom variablene og dermed ikke klarer å predikere med høy nøyaktighet. For å lage en modell som er generaliserbar for andre typer data, kan CV brukes for å dele opp datasettet i flere deler, slik at nye optimaliseringer ikke trener seg selv på den samme inndelingen av trening- og testsett i datasettet.

K-fold cross validation

K-fold CV er metoden jeg har brukt for denne studien, hvor både trening- og testsettet blir delt inn etter hvor mange versjoner av datasettet som ønskes å bli testet og validert. Berrar (2019) omtaler disse delene som k-folds. Tanken er at datasettet skal deles inn etter like mange tilfeldige deler som testes og valideres opp mot hverandre, slik at den samlede nøyaktigheten av alle delene reduserer støy i datasettet og gjør det mer nøyaktig uten å overfitte datasettet. Eksempelvis vil en mengde på 5 k-folds utgjøre at treningssettet blir delt inn i 5 grupper som hver tilsvarer 20% hver av treningssettet. Samtidig vil det samme skje med testsettet. Gjennomsnittet av CV-nøyaktigheten for alle 5 delmodellene gir en total

nøyaktighetsvurdering av datasettet, hvor den tilfeldige utvelgelsen av delmodellene unngår overfitting.

Stratifisering

Stratifisering er en metode innen k-fold CV som innebærer at hver k-fold/delmodell opprettholder samme dimensjon av klassifiseringer (Berrar, 2019). Praktisk eksempel kan være en modell som skal predikere om et bilde viser en bil eller båt, hvor modellen totalt sett predikerer at 70% av tilfellene er bil og 30% båt. Stratifisering betyr at denne 70/30-fordelingen opprettholdes i hver individuelle k-fold/delmodell.

Cross validation med andre algoritmer

For å vurdere hvordan SVM presterer i forhold til andre maskinlæringsalgoritmer, må nøyaktigheten kalkuleres sammenlignet med andre algoritmer. Utgangspunktet er i scikit-learn's cross_val_score-default-verdier (Pedregosa et al., 2011). Disse verdiene tilsvarer en k-fold på 5, hvor dette er tilfellet for de modellene hvor det ikke nevnes noe annet.

SVM skal sammenlignes med K-nearest neighbors, random forest og decision trees. Da denne masteroppgaven primært er basert rundt optimalisering av SVM, vil jeg ikke utdype innholdet bak de 3 andre algoritmene, da jeg kun er interessert i nøyaktigheten og hvordan algoritmene presterer på bakgrunn av CV.

Stacked ensemble learning

Stacked ensemble learning blir videre omtalt som stacking og innebærer en kombinasjon av de best presterende algoritmene, basert på CV-resultatene. Forklart annerledes vil stacking produsere en helt ny meta-modell på grunnlag av prediksjonsevnen til de andre algoritmene (Brownlee, 2021). Metamodellen er med andre ord basert på prediksjonsresultatene til de andre algoritmene og ikke basert på verdiene til det originale trening- og testsettet (Brownlee, 2021).

I tillegg til nøyaktighet og F1, måler stacked-modellen også Matthews korrelasjonskoeffisient, videre omtalt som MCC. MCC er bedre enn både nøyaktighet og F1, da den kun avgir en høy verdi dersom den predikerer alt riktig. Dette innebærer god prediksjonsevne for ekte positive, falske negative, virkelige negative og falske positive prediksjoner, ifølge Chicco & Jurman (2020).

3.4 Bias-variance tradeoff.

I dette delkapittelet forklares delene som utgjør bias-variance tradeoff innen maskinlæring og hvordan dette kan føre til enten overfitting eller underfitting.

Bias-variance tradeoff justerer bias og variance etter hva som er behovet per modell. Vanligvis vil en økning av den ene, føre til en reduksjon av den andre, derav navnet tradeoff. Parafraserer Gèron (2019, 162) som hevder at en økning av kompleksitet i en modell, tilsvarer økt variance og redusert bias, hvor det motsatte er tilfelle ved reduksjon av en modells kompleksitet.

Bias viser sammenhengen mellom feil i modellens prediksjon, sammenlignet med datasettet. Med andre ord i hvor stor grad treningssettet matcher opp mot testsettet (Wickramasinghe, 2021). Eksempelvis dersom observasjonene i testsettet matcher treningssettet tilnærmet perfekt, er det lite bias. Samtidig vil det være høy bias, hvis testsettet medfører mye feilklassifiseringer.

Variance er modellens evne til å reagere på endringer i treningssettet og hvor sensitiv modellen er ved små forskjeller i treningssettet (Gèron, 2019). Wickramasinghe (2021) mener høy variance ikke er ønskelig, da det kan tyde på mye kompleks støy i datasettet og antydning til overfitting. Eksempelvis vil en høy variance føre til forskjellige resultater fra testsett til testsett, hvor prediksjonsraten ikke er konsistent og heller ikke generaliserbar til andre datasett.

Mean squared error, videre omtalt som MSE er også et mål som tolkes i henhold til bias-variance tradeoff. Denne verdien vektlegger avstanden mellom observasjonene og regresjonslinja, hvor en økt MSE tilsvarer dårligere prediksjon eller klassifisering. Redusert MSE er ønskelig, da dette betyr færre feil og en modell som gir bedre prediksjoner eller klassifiseringer (Glen, u.å).

Ifølge Doroudi (2020) vil bias-variance trade-off innen maskinlæring blant annet fokusere på om datasettet er preget av underfitting eller overfitting. Parafraserer Gèron (2019, 30) som mener at underfitting oppstår i modeller som ikke fanger opp kompleksiteten av datasettet og av den grunn ikke klarer å predikere eller klassifisere riktig. Gèron (2019,30) foreslår at en legger til flere variabler, bruker andre optimaliseringstester innen FS for å finne de beste variablene eller tuner hyperparameterne for å motvirke underfitting. Overfitting er fenomenet hvor modellen matcher treningssettet i høy grad, men hvor modellen ikke er generaliserbar (Gèron, 2019, 28). Forklart annerledes er datasettet og modellen såpass godt tilpasset det unike datasettet, hvor samme fremgangsmåte ikke er overførbart for andre datasett.

4. Data

Her vil det gjøres rede for begrensninger ved datasettet, i tillegg til både den avhengige og de uavhengige variablene for første SVM-modell og de variablene som gjenstod etter optimaliseringsprosessen. Utgangspunktet for valg av variabler baserte seg på en datacleaning-prosess hvor jeg inkluderte alle variabler fra ulike excel-dokumenter som hadde en sammenheng med plukkaktiviteten fra et lager, hvor observasjonene hadde få nullverdier. Variablene skal ha som hensikt å forklare variansen i tiden det tar å plukke en konkret mengde ordrelinjer.

4.1 Forklaring og utvelgelse av variabler

Metodene baserer seg på optimaliseringsteknikker som regresjon, variance threshold og pearsons korrelasjonskoeffisient. Regresjonens hensikt er å avdekke signifikante p-verdier for å vise hvilke uavhengige variabler som har en sterk korrelasjon og forklaringskraft av plukktiden på antall ordrelinjer. Variance threshold fjerner de variablene som forklarer kun en bestemt prosentandel av variansen i y. Jeg valgte å sette grensen på 0% for å kun fjerne de variablene som ikke forklarte noe av variansen. Resultatet av dette var å fjerne én ytterligere variabel. Pearsons korrelasjonskoeffisient fant ingen kovariasjon mellom de uavhengige variablene, hvor det gjenstod 10 uavhengige variabler. I de følgende delavsnittene skal jeg gjøre rede for både den avhengige og de uavhengige variablene.

ACT_WEIGHT: Varenes totale vekt per enhet.

Varens totale vekt for et sett med ordrelinjer ønsker å måle om høyere vekt tilsvarer lenger plukktid eller ikke. Høyere vekt trenger ikke nødvendigvis å tilsvare lenger plukktid, spesielt ikke om det er helpaller som plukkes, hvor varene er forhåndsplastret og det plukkes med truck. Uansett vil det undersøkes om vekten påvirker plukktiden.

ACT_CARTON: Antall enheter per kolli som er sendt per ordre

Antall kolli sendt per ordre indikerer pakkearbeid for å ferdigstille en konkret ordre. Når et sett varer blir plukket, blir de vanligvis pakket videre i andre esker. Det variabelen ikke sier oss, er størrelsen på kartongen som blir brukt, men kun forholder seg til antallet. Variabelen

vil på en annen side gi en indikasjon på hvor mye arbeid som måtte legges ned i den avsluttende fasen av arbeidet.

FROM_ITEM: Konkret vare

Denne variabelen tydeliggjør hvilken vare som er blitt bestilt. Da jeg ønsker å predikere om plukktiden er det nødvendig å påpeke om det er enkelte produkter som påvirker plukktiden eller ikke.

FROM_LOCATION: plukklokasjon

Plukklokasjonen tilsvarer de unike plukklokasjonene på lageret, hvor varer automatisk bufres av et automatisk høylager som fyller på varer etter behov. Ettersom målet er å finne ut av hva som fører til variasjon i plukktiden av ordrelinjer, er det vesentlig å finne ut om unike plukklokasjoner påvirker plukktiden.

FROM_ZONE: plukkområde

Plukkområdet skiller hovedsakelig mellom plukk med T1-truck fra helpallelokasjoner og kasselokasjoner. For å avgjøre om konkrete plukkområder påvirker plukktiden, er dette en viktig uavhengig variabel.

TO_ZONE: plukkområde helpaller

Plukkområdet omfatter hovedsakelig helpalleplukk, hvor det hadde vært interessant å se om plukk fra de forskjellige områdene påvirket plukktiden per sett med ordrelinjer.

FROM_CENTER: Lagerområde

Det skilles mellom 3 plukkområder, hvor det hovedsakelig plukkes fra helpalle- og kasselokasjoner langs bakken med T1-truck som deretter tar med varene til pakking. Hvorvidt forskjeller på plukkområder påvirker plukktiden når plukkprosessen er tilsvarende lik for alle plukkområder er nok lite sannsynlig. Men det kan oppstå noe variasjon for noen ordre som må

bli ferdigstilt på forskjellige måter. Eksempelvis om et sett ordrelinjer må pakkes om til flere paller, noen små kolli eller kun leveres som helpaller, utgjør en forskjell i tidsutførelsen. Da vil det være interessant å avdekke trender som skjer ved forskjellige plukkområder, dersom dette igjen påvirker plukktiden.

FROM_QTY: Antall enheter bestilt

Antall enheter bestilt viser kvantiteten av varer som er bestilt. Sannsynligvis vil et større kvantum av varer føre til lengre plukktid, men ikke nødvendigvis. Dette er blant annet avhengig av hvor mange enheter det er per pakkeenhet, hvor raskt det er mulig å plukke riktig antall eller plukklokasjon/plukkområde. Derfor vil det være omdiskutert om antall enheter bestilt innehar sterk forklaringskraft i forhold til plukktiden.

FROM_UOM: Pakkeenhet

Pakkeenheten tydeliggjør hvordan et sett med enheter av varen blir oppbevart. Dette kan eksempelvis være i kartonger, esker, stykkevis, poser eller kanner, for å nevne noen. Hvorvidt disse enhetene påvirker plukktiden vil være interessant å undersøke, dersom det er tydelige forskjeller på plukktiden fra pakkeenhet til pakkeenhet.

ORD_LINE_NO: Antall ordrelinjer

Denne variabelen viser hvor mange ordrelinjer som er bestilt per vare i en konkret ordre. Jeg ønsker blant annet å undersøke om flere ordrelinjer fører til lengre plukktid. Av den grunn er hensiktsmessig å inkludere denne variabelen.

Tid_lengde_tidsintervall: Tid for å ferdigplukke sett med ordrelinjer (Avhengig variabel (y))

Tid for å ferdig plukke en ordre er justert delvis eksponentielt og målt i minutter. Starter fra 1 minutt, til 2 minutter, 5 minutter, 10 minutter og videre helt opp til 24 timer. Tiden er den avhengige variabelen, hvor vi ønsker å undersøke om de tidligere nevnte variablene påvirker plukktiden og hvorvidt SVM-modellen er egnet for dette.

5. Analyse og resultater

Dette kapittelet skal vise hvordan analysen er gjennomført og de tilhørende resultatene. Først skal den første SVM-modellens oppbygning og resultater gjøres rede for. Deretter skal optimaliserings-resultatene gjøres rede for. Disse er henholdsvis multippel lineær regresjon, variance threshold og pearsons korrelasjonskoeffisient. Neste delkapittel vil omhandle konstruksjonen av den andre SVM-modellen som tar hensyn til resultatet fra optimaliseringen. Videre skal resultatene fra CV med ulike algoritmer sammenlignes, hvor CV-resultatene avslutningsvis bidrar til å konstruere en helt ny prediktiv modell ved hjelp av stacking.

5.1 SVM-modell 1

Den første SVM-modellen viser hvilke verdier som er valgt for C, Gamma, type kernel, accuracy- og F1-score og resultatet av parametertuning. Den første SVM-modellen inneholder 15 uavhengige variabler. I tillegg bestod treningssettet av 70% og testsettet av 30%.

C, Gamma og kernel

Opprinnelig C er lik 1 og gamma er lik 0,5. Disse verdiene tilsvarer standardverdiene til C og gamma i SVM-dokumentasjonen til Pedregosa et al. (2011), som er maskinlæringspakken som ble brukt til denne analysen. I tillegg blir kernel satt til radial basis function, videre omtalt som RBF, hvor både trenings- og testsett for X blir skalert for at SVM skal kunne vurdere avstanden mellom observasjoner (Toku, 2021).

Accuracy og F1

Tabell 2: Accuracy og F1-score for første SVM-modell

```
Accuracy (RBF Kernel):  37.50
F1 (RBF Kernel):  23.19
```

Resultatet av nøyaktigheten og F1-scoren til den første SVM-modellen tilsier at modellen er lite nøyaktig. Ifølge nøyaktigheten vil 37,5% av tilfellene predikere plukktiden på et sett

ordrelinjer riktig. F1 justerer nøyaktigheten opp mot antall feilklassifiseringer også, noe som reduserer nøyaktigheten av modellen ytterligere.

Hyperparametertuning

For å finne de optimale parameterne av kernel, C og gamma, ble det gjennomført en parametertuning ved hjelp av pakken GridSearchCV i scikit-learn (Pedregosa et al., 2011). Resultatene er fremstilt i tabell 3 nedenfor. Parametertuningen tar hensyn til ulike verdier av C og gamma, hvor RBF ble satt som standard kernel. Hensikten er å finne kombinasjonen av verdier som får modellen til å prestere best. Resultatet viser at når C-parameteren er lik 1 og kernel er lik RBF, vil SVM-modellen prestere best. Dette er en liten og neglisjerbar forbedring i forhold til den opprinnelige scoren som var litt mindre enn forbedringen ved parametertuning. Da C var lik 1 i resultatet for første SVM-modell, er det åpenbart at en liten endring i gamma har forårsaket endringen i scoren.

Tabell 3: Hyperparametertuning for første SVM-modell

	param_C	param_kernel	mean_test_score
0	1	rbf	0.402215
1	10	rbf	0.383818
2	20	rbf	0.374709
3	100	rbf	0.357204

5.2 Optimalisering/Feature selection

Regresjon

Regresjon brukes for å sjekke forklaringskraften til de uavhengige variablene og hvorvidt de har noe korrelasjon med den avhengige variabelen. Hensikten er å finne og fjerne de variablene som ikke forklarer noe av endringen i plukktiden av et sett ordrelinjer.

Tabell 4: Regresjon med alle variabler

Out[148]:

OLS Regression Results

Dep. Variable:	tid_lengde_tidsintervall	**R-squared:**	0.019
Model:	OLS	**Adj. R-squared:**	0.017
Method:	Least Squares	**F-statistic:**	9.507
Date:	Thu, 07 Jul 2022	**Prob (F-statistic):**	2.79e-21
Time:	15:12:33	**Log-Likelihood:**	-46470.
No. Observations:	7000	**AIC:**	9.297e+04
Df Residuals:	6985	**BIC:**	9.307e+04
Df Model:	14		
Covariance Type:	nonrobust		

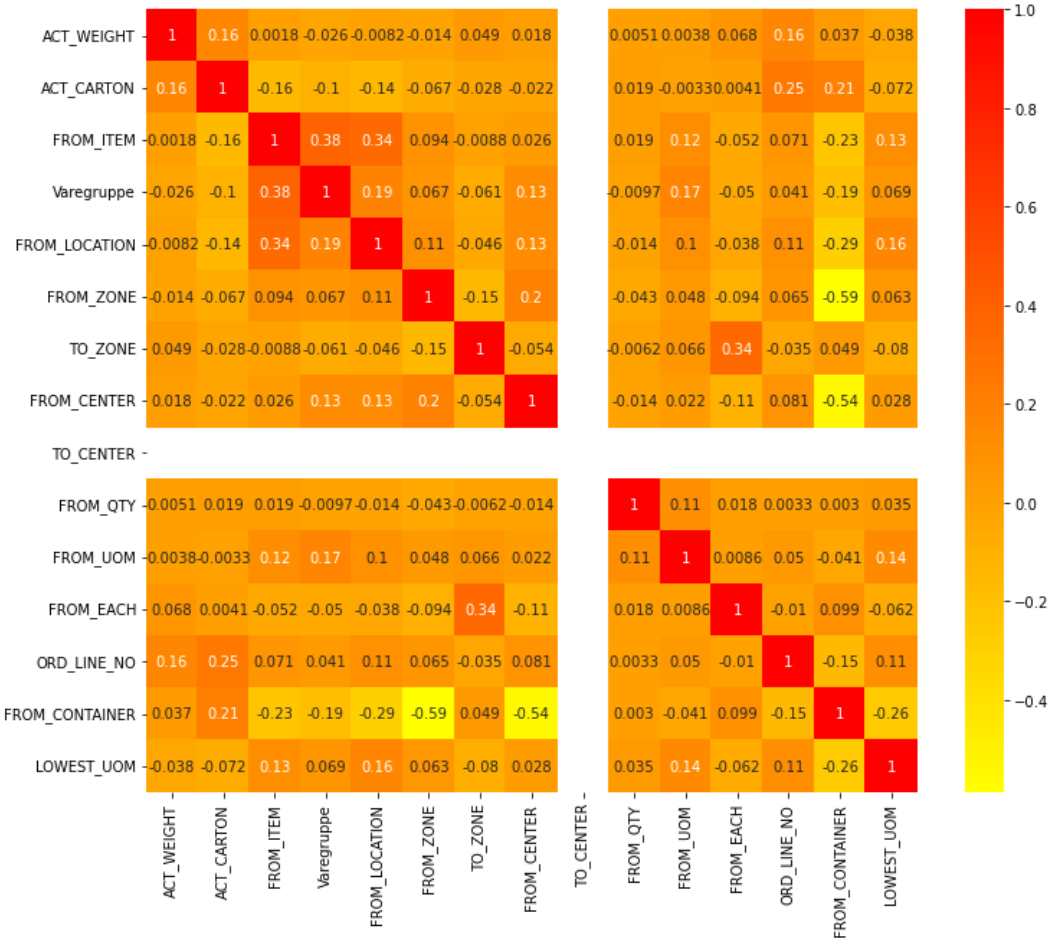
	coef	std err	t	P>|t|	[0.025	0.975]
const	122.2490	16.474	7.421	0.000	89.955	154.543
ACT_WEIGHT	0.0059	0.003	1.760	0.078	-0.001	0.013
ACT_CARTON	-0.8673	0.371	-2.339	0.019	-1.594	-0.141
FROM_ITEM	0.0293	0.007	4.021	0.000	0.015	0.044
Varegruppe	0.0126	0.042	0.300	0.765	-0.070	0.095
FROM_LOCATION	0.0125	0.008	1.600	0.110	-0.003	0.028
FROM_ZONE	0.8385	0.428	1.960	0.050	-0.000	1.677
TO_ZONE	49.0295	7.010	6.995	0.000	35.288	62.770
FROM_CENTER	10.8433	4.253	2.550	0.011	2.506	19.181
TO_CENTER	2.146e-13	3.1e-14	6.927	0.000	1.54e-13	2.75e-13
FROM_QTY	0.1170	0.075	1.558	0.119	-0.030	0.264
FROM_UOM	-1.3501	0.764	-1.767	0.077	-2.848	0.148
FROM_EACH	0.0016	0.001	1.519	0.129	-0.000	0.004
ORD_LINE_NO	-0.0161	0.037	-0.432	0.666	-0.089	0.057
FROM_CONTAINER	-0.3712	4.184	-0.089	0.929	-8.573	7.831
LOWEST_UOM	0.4815	0.579	0.831	0.406	-0.654	1.617

Omnibus:	7076.878	**Durbin-Watson:**	1.983
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	337763.637
Skew:	5.126	**Prob(JB):**	0.00
Kurtosis:	35.449	**Cond. No.**	1.16e+19

Fra tabell 4 ovenfor ovenfor vises både r-kvadrat og justert r-kvadrat som tilnærmet lik null, noe som betyr at datasettet forklarer nesten ingenting av variansen i plukktiden. Samtidig kan signifikante p-verdier indikere at det fortsatt er en sammenheng mellom de uavhengige og den avhengige variabelen (Minitab, 2014). Av den grunn vil alle variabler som har en p<0,120 beholdes. Denne grensen velges fordi da jeg blant annet ønsker å undersøke om konkrete plukklokasjoner påvirker plukktiden. Resultatet av dette ble en fjerning av 5 variabler.

Pearsons R

Pearsons korrelasjonskoeffisient testet for kovariasjon mellom de uavhengige variablene. Alle 15 variabler ble testet for kovariasjon, hvor tilnærmet verdi lik 1 tilsier at variablene har perfekt kovarians, der en lav verdi indikerer lav kovarians. Hensikten er å fjerne variabler som forklarer det samme. Resultatet var at ingen var preget av kovariasjon, hvor ingen ble fjernet for å optimalisere algoritmen.



Figur 4: Test for Pearsons korrelasjonskoeffisient

VarianceThreshold

Variance threshold har som hensikt å fjerne variablene som ikke forklarer noe av variansen i den avhengige variabelen. Variansegrensen ble satt til 0, da jeg kun ønsket å fjerne de variablene som ikke forklarte noe av variansen i y. Grensen ble satt på bakgrunn av

regresjonen, da modellen hadde liten til ingen forklaringskraft uansett, hvor jeg kun ville fjerne de som ikke ga noe merverdi til modellen. Resultatet ble fjerning av en ytterligere variabel (TO_CENTER) som ikke forklarte noe av variansen i den avhengige variabelen.

5.3 SVM-modell 2

På grunnlag av resultatene fra å teste regresjon, Pearsons R og Variancethreshold, kan SVM-modellen trenes på nytt, med færre variabler og mindre støy i datasettet. I likhet med den første modellen, ble C satt som 1 og gamma som 0,5. Treningssettet var 70% og testsettet 30%.

Accuracy og F1

Med mindre støy, presterte andre SVM-modell bedre enn den første. Nøyaktighet justert mot antall feilklassifiseringer i F1, viste en nøyaktighet til å predikere plukktiden med ca. 55% nøyaktighet. Noe som er en betydelig forbedring i forhold til 23% fra den første modellen. 55% er på en annen side ikke en god prediksjonsnøyaktighet.

Tabell 5: Accuracy og F1-score for andre SVM-modell

```
Accuracy (RBF Kernel):  56.52
F1 (RBF Kernel):  54.43
```

Hyperparametertuning

I motsetning til forrige modell, er det større forskjell på hvor godt modellen presterer, ved parameterjustering som kan ses i tabell 6 nedenfor. For denne SVM-modellen er det en forbedring på ca. 15% dersom en justerer C fra 1 til 100.

Tabell 6: Hyperparametertuning for andre SVM.modell

	param_C	param_kernel	mean_test_score
0	1	rbf	0.391633
1	10	rbf	0.453469
2	20	rbf	0.482245
3	100	rbf	0.546939

5.4 Cross validation

Resultatene fra CV i tabellen under regnet ut en CV-score for algoritmene SVM, Random forest (RF), K-nearest neighbours (KNN) og Decision trees (DT). CV-scorens hensikt er å vise hvor nøyaktige de ulike algoritmene er til å predikere. Alle algoritmene trente på 5 forskjellige k-folds, hvor hver av disse 5 delene utregnet 5 nøyaktighetsverdier som trente på ulike deler av treningssettet. 1 representerer perfekt prediksjon, hvor 0 betyr ingen riktige predikerte verdier.

Tabell 7: Cross Validation score

CV-score					
SVM	0,3328	0,3392	0,3342	0,335	0,335
RF	0,5942	0,6	0,6021	0,5892	0,5814
KNN	0,3907	0,385	0,3907	0,3742	0,3892
DT	0,575	0,5957	0,5957	0,5664	0,5621

Modellen ovenfor viser resultatene fra CV, hvor RF så vidt er mer nøyaktig enn DT. Dette betyr at RF er ca. 60% nøyaktig i sin prediksjon av plukktid av et sett ordrelinjer. Samtidig presterer SVM dårligere enn SVM-modell 2.

5.5 Stacking ensemble method

Resultatet av stacking, basert på SVM, RF, KNN og DT, resulterte i en ny metamodell som var ca. 56% nøyaktig i sin prediksjonsevne. Den opprettholder tilsvarende prediksjonsevne

justert mot feilklassifiseringer i F1-scoren. MCC er redusert da den har avdekket ytterligere feilklassifiseringer som senker nøyaktigheten. Metamodellen ca. 42% nøyaktig som ellers ikke er en god prediksjonsevne.

Tabell 8: Accuracy, MCC og F1-score for trenings- og testsettet i stacking-modell 1

```
Model performance for Training set
- Accuracy: 0.9851020408163266
- MCC: 0.9809906717248442
- F1 score: 0.9851019553761671
---------------------------------
Model performance for Test set
- Accuracy: 0.5614285714285714
- MCC: 0.42233008637420005
- F1 score: 0.548040594796685
```

Metamodellen hadde høy bias og lav variance. Gutta (2020) hevder at denne kombinasjonen tyder på at prediksjonene er unøyaktige, men samtidig konsistente. Gutta (2020) mener at høy bias og lav variance antyder underfitting, hvor det er brukt for få variabler for å utarbeide modellen. Da bias er betydelig høyere enn variance, indikerer dette at bias bør reduseres ved å inkludere flere variabler eller bruke andre modeller (Gutta, 2020). I tillegg innehar modellen en kvadrert error-verdi som måler hvor mye støy som er i datasettet (Gutta, 2020). Denne verdien tilsvarte 23692.825 som indikerer at det er mye støy i datasettet som påvirker prediksjonsevnen til modellen. Dette gjenspeiler seg i prediksjonsevnen til metamodellen, da den ikke er nøyaktig.

Tabell 9: Bias, variance og MSE i stacking-modell 1

Average bias	87951820,926
Average variance	3100,665
Mean squared error	23692,825

5.6 Regresjon og stacking-modell 2

For å prøve å optimalisere stacking-modellen ble det kun inkludert de variablene som hadde en signifikant p-verdi på 95% signifikansnivå fra regresjonen. Utenom TO_CENTER som ble fjernet av Variancethreshold, var det kun 4 variabler som ble brukt: ACT_CARTON, FROM_ITEM, FROM_ZONE og FROM_CENTER.

Regresjonen for den andre stacking-modellen viser i likhet med den første regresjonen at noen av variablene har signifikant korrelasjon med plukktiden, men nesten ingen forklaringskraft i form av lavt r-kvadrat. I tillegg er AIC høy og tilsvarende lik den første regresjonen, noe som tyder på at den ikke er statistisk signifikant bedre enn den andre. Den er med andre ord ikke bedre, selv om ytterligere variabler og støy er blitt fjernet.

Tabell 10: Regresjon for stacking-modell 2

OLS Regression Results

Dep. Variable:	tid_lengde_tidsintervall	R-squared:	0.008
Model:	OLS	Adj. R-squared:	0.007
Method:	Least Squares	F-statistic:	14.13
Date:	Sun, 17 Jul 2022	Prob (F-statistic):	1.74e-11
Time:	15:59:18	Log-Likelihood:	-46508.
No. Observations:	7000	AIC:	9.303e+04
Df Residuals:	6995	BIC:	9.306e+04
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	129.1543	7.179	17.991	0.000	115.082	143.227
ACT_CARTON	-0.9914	0.343	-2.887	0.004	-1.665	-0.318
FROM_ITEM	0.0327	0.006	5.093	0.000	0.020	0.045
FROM_ZONE	0.4528	0.337	1.342	0.180	-0.209	1.114
FROM_CENTER	10.5768	3.474	3.044	0.002	3.766	17.388
TO_CENTER	0	0	nan	nan	0	0

Omnibus:	7115.978	Durbin-Watson:	1.986
Prob(Omnibus):	0.000	Jarque-Bera (JB):	343464.314
Skew:	5.170	Prob(JB):	0.00
Kurtosis:	35.721	Cond. No.	inf

Igjen, ble det utarbeidet en metamodell basert på SVM, RF, KNN og DT. Denne modellen var mer nøyaktig på ca. 83% i henhold til både nøyaktighet og F1-score. MCC som justerer mot feilklassifiseringer er tilsvarende like høy på ca. 78%, noe som er over dobbelt så bra som forrige metamodell.

Tabell 11: Accuracy, MCC og F1-score for stacking-modell 2

```
Model performance for Training set
- Accuracy: 0.9424489795918367
- MCC: 0.9270360769901892
- F1 score: 0.9427814142928954
---------------------------------
Model performance for Test set
- Accuracy: 0.8385714285714285
- MCC: 0.7927723366712076
- F1 score: 0.837689332810458
```

Den nye metamodellen har høy bias og lav variance relativt sett og følger samme trend som forrige metamodell, men har lavere error-verdi som kan ses i tabellen nedenfor. Dette tyder på at jeg har redusert litt støy i datasettet, selv om det fortsatt ikke tilstrekkelig godt nok. Totalt indikerer verdiene at modellen er preget av underfitting på bakgrunn av kun de 4 variablene som modellen baserer seg på. Med andre ord er plukktiden av et sett ordrelinjer for komplisert å predikere med disse 4 variablene.

Tabell 12: Bias, variance og MSE for stacking-modell 2

Average bias	124397672,988
Average variance	6377,628
Mean squared error	14385,797

6. Diskusjon

Dette kapittelet skal gå gjennom resultatene og drøfte funnene i relasjon til hypotesene i kronologisk rekkefølge. Først hypotese 1 og 2 hver for seg, hvor det avsluttes med hypotese 4 og 5 sammen.

6.1 Hypotese 1

Hypotese 1: SVM er ikke en egnet algoritme til å optimalisere kostnadsdrivere.

Ved bruk av hyperparametertuning for SVM var hverken den første eller andre SVM-modellen særlig nøyaktig i sine prediksjoner av plukktiden av et sett ordrelinjer. Den andre modellen ble forbedret med ca. 15% nøyaktighet etter FS-optimalisering og økte til ca. 54% nøyaktighet. Sagt annerledes hadde ingen av SVM-modellene god prediksjonsevne.

Samtidig viser regresjonen for første SVM-modell i tabell 4 at justert r-kvadrat er tilnærmet lik null, selv om noen variabler har signifikant p-verdi og at AIC har en høy verdi på ca. 9,3. Dette er likt for andre SVM-modell som har tilsvarende lav justert r-kvadrat og nesten identisk AIC. Den lave forklaringskraften i begge modellene, tyder på at variablene som er brukt ikke passer seg for å forklare variasjon i plukktiden av et sett ordrelinjer. De høye AIC-verdiene indikerer at ingen av SVM-modellene passer bra til prediksjon av plukktid og hvor ingen av SVM-modellene er bedre enn den andre, da det ikke er en signifikant forskjell på AIC-verdiene er tilsvarende like (Bevans, 2020). I tillegg tyder modellen at det er sterk korrelasjon mellom konkrete varer og plukkområder i forhold til hvor lang tid det tar å plukke et par ordrelinjer, men lav justert r-kvadrat understreker null årsakssammenheng.

CV viste at ingen av de andre algoritmene presterte bedre enn SVM. Noe som tyder på at unøyaktigheten ikke tilegnes til SVM alene, hvor datasettets lave forklaringskraft vises uavhengig av hvilken algoritme som ble brukt.

Totalt sett er ikke denne analysen nok til å avskrive SVM som egnet til å optimalisere kostnadsdrivere eller ikke, men er noe som må undersøkes nærmere i videre forskning. Muligens med fremgangsmetoden til Kim & Han (2003) hvor en evolusjonær algoritme blir foreslått som en komplementær optimaliseringsmetodikk.

6.2 Hypotese 2

> **Hypotese 2**: Andre algoritmer presterer bedre enn SVM ved å predikere plukktiden for et sett ordrelinjer.

For å avgjøre om andre maskinlæringsalgoritmer presterer bedre enn SVM for å predikere plukktiden på et sett ordrelinjer, var det hensiktsmessig å gjennomføre CV. Tidligere forskning har ikke nødvendigvis brukt denne fremgangsmetoden for å kombinere og sammenligne ulike algoritmer, men som viste seg å være en aktuell fremgangsmåte da SVM ikke viste til nøyaktig prediksjonsevne.

Alle de andre algoritmene predikerte plukktiden mer nøyaktig enn SVM, som et resultat av CV. På en annen side viste andre SVM-modell en tilsvarende lik nøyaktighet på 54% som den beste algoritmen på bakgrunn av CV som var RF på 60%. Grunnen til diskrepansen i ytelsen til SVM er mangelen på hyperparameterjustering i CV. Dette betyr også at de andre algoritmene kunne blitt forbedret ytterligere ved hyperparameterjustering per algoritme, men dette var utenfor denne masteroppgavens omfang.

Grunnen til at ingen av modellene er mer nøyaktige enn 60% kan forklares delvis av resultatene fra regresjonen. Forklaringskraften til modellen i form av nær null justert r-kvadrat, tyder på at variablene ikke passer til å predikere plukktiden av et sett ordrelinjer. Hvor en bedre prediktiv nøyaktighet kun hadde indikert at modellen var preget av overfitting. Dette var tilfellet med den siste stacking-modellen som var omtrent 80 % nøyaktig, men som indikerte at prediksjon av plukktiden er mer komplisert enn modellen jeg utarbeidet.

I motsetning til Kim & Han (2003), brukte jeg CV og stacking istedenfor en genetisk algoritme i kombinasjon med SVM eller noen av de andre algoritmene. Da ingen av SVM- eller stacking-modellene, samt resultat fra CV tydet på god forklaringskraft i datasettet, ville en genetisk algoritme mest sannsynlig ført til mer overfitting og mindre sjanse for at modellen kunne generaliseres. Dette underbygges av det lave r-kvadratet og AIC i hver regresjon som understreker at ingen av variablene forklarer endringen i plukktiden av et sett ordrelinjer.

6.3 Hypotese 3 og 4

> **Hypotese 3**: SVM/Stacking-modellene kan predikere forskjell i plukktiden for forskjellige varer.
>
> **Hypotese 4**: SVM/Stacking-modellene kan predikere forskjell i plukktiden for forskjellige plukkområder.

Plukktiden på et sett ordrelinjer kan ikke påvises å være en god kostnadsdriver i denne studien. Da en plukkliste kan inneha en ordrelinje med alt fra en til hundrevis av enheter, vil det være store forskjeller på hvor lang tid det tar å plukke per ordrelinje.

Tabell 10 viser siste regresjon for modellen med 95% signifikansnivå, hvor det er en sterk korrelasjon mellom varene og plukktiden av ordrelinjer, men ikke plukkområdet. Samtidig er også forklaringskraften til denne modellen tilnærmet null med en lav justert r-kvadrat. Totalt indikerer dette at hverken konkrete varer, plukkområder eller lagerområder har en årsakssammenheng med plukktiden av et sett ordrelinjer.

I tillegg viser hverken regresjonen i tabell 4 eller 10 at det er noen av variablene som forklarer endringen i plukktiden av et sett ordrelinjer. Kombinert med relativt lave prediksjonsverdier fra alle modellene, vil det tilsi at det ikke er nødvendig å utvikle forskjellige kostnadsdrivere for plukkaktiviteten på lageret, basert på forskjellige plukkområder, varer eller noen av de andre uavhengige variablene.

7. Konklusjon

Formålet med masteroppgaven var å optimalisere en kostnadsdriver innen ABC ved hjelp av maskinlæring. Mer konkret om SVM nøyaktig kunne predikere plukktiden av et sett ordrelinjer og hvorvidt andre maskinlæringsalgoritmer presterte bedre enn en optimalisert SVM-algoritme. I tillegg undersøkte studien om konkrete varer eller plukkområder forklarte forskjellen i plukktid.

Datasettet har basert seg på variabler for plukkaktivitet, vareinformasjon og annen metadata om forsendelser over en tidsperiode på ca. 3 måneder for et disponibelt lager fra samarbeidende bedrift.

Selv om ingen av SVM-modellene hadde god prediksjonsevne, var dette også tilfelle for de andre algoritmene i tillegg til stacking-modellene. I kombinasjon med lav justert r-kvadrat for begge regresjonene, tyder det på at jeg ikke kan konkludere at SVM kan avskrives som en algoritme som ikke kan optimalisere kostnadsdrivere.

Den dårlige prediksjonsevnen til begge SVM-modellene førte til valget for å gjennomføre en CV for å avgjøre om andre algoritmer viste en høyere prediktiv nøyaktighet. RF viste en minimal økt forbedring i forhold til den andre SVM-modellen som hadde blitt optimalisert gjennom hyperparametertuning og FS, hvor RF fortsatt ikke hadde god nok nøyaktighet på 60%. I tillegg viste regresjonen og resultatene fra bias/variance at modellen ikke var god nok til å predikere plukktiden av et sett ordrelinjer. Dermed har jeg ikke tilstrekkelig grunnlag for å konkludere om andre algoritmer presterer bedre enn SVM ved prediksjon av plukktid.

Mangelen på forklaringskraft i modellene understreker at både varer og plukkområder ikke alene forklarer endringen i plukktid av et sett ordrelinjer, selv om det finnes en statistisk signifikant korrelasjon. I tillegg viser bias og variance at modellene enten er preget av for mye støy eller underfitting, hvor AIC understreker at ingen av modellene er godt tilpasset datasettet og ikke skiller seg statistisk signifikant fra hverandre. Samlet vil konklusjonen for

hypotese 3 og 4 være at hverken varer eller plukkområder alene, ikke forklarer endringen i plukktiden av et sett ordrelinjer. Med andre ord vil det ikke være nødvendig å komplisere antall ordrelinjer som kostnadsdriver for plukk, da studien ikke har funnet noe som statistisk signifikant påvirker plukktiden.

Om SVM eller noen av de andre maskinlæringsalgoritmene kan gi merverdi til ABC ved optimalisering av kostnadsdrivere er fortsatt uavklart. Istedenfor å kun analysere reelle datasett, kan videre forskning ta utgangspunkt i simulert data som i Kim & Han (2003) og Babad & Balachandran (1993) for å utarbeide en mer generaliserbar modell. Begge sistnevnte artikler har vist at maskinlæring kan brukes for å optimalisere kostnadsdrivere med ulik metodikk. Forslaget til Kim & Han (2003) om å kombinere en genetisk algoritme med SVM er noe som burde prøves i videre forskning.

8. Begrensninger

En begrensning ved datasettet var prosessdataene som omfattet 46 arbeidsprosesser, hadde varierende antall observasjoner. Noen arbeidsaktiviteter hadde tilnærmet ingen observasjoner, hvor andre hadde over 20 000 observasjoner. Derfor ble det avgjort å kun sette søkelys på plukkaktivitet med antall ordrelinjer som kostnadsdriver, da denne aktiviteten hadde tilstrekkelig nok observasjoner på ca. 25 000 etter data-cleaning. I tillegg måtte disse downsamples og stratifiseres ytterligere ned til 7000 datapunkter for å redusere treningstiden. SVM trenger ikke like mange observasjoner for å trene modellen, hvor andre modeller trenger betydelig flere observasjoner. Dette kan være en fordel for treningen av SVM-modellene, hvor de andre modellene utført i CV kan være preget av underfitting. På en annen side er det ingen konkret regel på hvor mange observasjoner en trenger for å trene en god maskinlæringsmodell, hvor det er viktig å se på korrelasjonen mellom variablene i datasettet (Sevey, 2017)

Andre begrensning var tidshorisonten på ca. 3 måneder, som satte restriksjoner på antall observasjoner som var tilgjengelig for å analysere.

Siste begrensning var antall variabler som kunne brukes i sammenheng med plukkaktivitet. Utførelsen av en tidkrevende data-cleaning prosess avdekket omtrent 15 variabler som kunne inkluderes uten at et omfattende antall nullverdier preget datasettet. På en annen side er denne begrensningen en mulig fallgruve hvis det oppstår menneskelige feil under data-cleaning prosessen, hvor aktuelle variabler utelates. Ved fremtidig forskning bør datasettet inneholde flere variabler med høyere forklaringskraft enn tilfellet for denne masteroppgaven.

9. Videre forskning

Som nevnt tidligere vil det være aktuelt å heller kombinere en genetisk algoritme med SVM ved optimalisering av kostnadsdrivere, som foreslått av Kim & Han (2003). Foreslår simulert data for å lage en generaliserbar modell og for å unngå mulig overfitting.

Dersom en velger å bruke samme metodikk som i denne masteroppgaven ved å gjennomføre CV, kan modellene følges opp tettere med ytterligere optimalisering av disse algoritmene.

Avslutningsvis vil det være viktig med både flere observasjoner over et lengre tidsrom og antall variabler som måler fenomenet. Dette er kontekstavhengig etter hvilken algoritme som brukes og hva som skal analyseres, men vanligvis krever maskinlæring mye data for å nøyaktig kunne predikere eller klassifisere godt nok.

Referanser

Babad, Y. M. & Balachandran, B. V. (1993). Cost Driver Optimization in Activity-Based Costing. *The Accounting Review*, 68 (3): 563-575.

Berrar, D. (2019). Cross-Validation. 542-545.

Bevans, R. (2020). *Akaike Information Criterion | When & How to Use It (Example)*. Tilgjengelig fra: https://www.scribbr.com/statistics/akaike-information-criterion/#:~:text=The%20AIC%20function%20is%202K,it%20is%20being%20compared%20to (lest 29.07.2022).

Bjørndal, M., Bjørnenak, T. & Johnsen, T. (2003). *Aktivitetsbasert kalkulasjon for regulerte tjenester - Erfaringer, prinsipielle retningslinjer og mulig anvendelse for nettvirksomhet i kraftsektoren.* Utvikling av en bedriftsøkonomisk kostnadsmodell for nettvirksomhet som basis for inntektskontroll. Tilgjengelig fra: https://openaccess.nhh.no/nhh-xmlui/bitstream/handle/11250/165251/R33_03.pdf?sequence=1&isAllowed=y (lest 14.06.2022).

Bjørnenak, T. (1993). ABC–hva er D? Grunnleggende prinsipper i aktivitetsbasert kalkulasjon. ABC-whats is D.

Brownlee, J. (2016 (a)). *Gentle Introduction to the Bias-Variance Trade-Off in Machine Learning*. Tilgjengelig fra: https://machinelearningmastery.com/gentle-introduction-to-the-bias-variance-trade-off-in-machine-learning/ (lest 14.06.2022).

Brownlee, J. (2016 (b)). *Support Vector Machines for Machine Learning*. Tilgjengelig fra: https://machinelearningmastery.com/support-vector-machines-for-machine-learning/ (lest 12.07.2022).

Brownlee, J. (2017). *Why One-Hot Encode Data in Machine Learning?* Tilgjengelig fra: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/ (lest 12.07.2022).

Brownlee, J. (2021). *Essence of Stacking Ensembles for Machine Learning*. Tilgjengelig fra: https://machinelearningmastery.com/essence-of-stacking-ensembles-for-machine-learning/.

Chandrashekar, G. & Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, 40 (1): 16-28. doi: https://doi.org/10.1016/j.compeleceng.2013.11.024.

Chen, M. (2008). *Greedy Algorithms*. A Greedy Algorithm with Forward-Looking Strategy.: INTECH Open Access Publisher.

Chicco, D. & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21 (6). doi: https://doi.org/10.1186/s12864-019-6413-7.

Cokins, G. & Capusneanu, S. (2010). Cost Drivers: Evolution and Benefits. *Theoretical and Applied Economics*, 17 (8): 7-16.

Cooper, R. & Kaplan, R. S. (1988). Measure Costs Right: Make the Right Decisions. *Harvard Business Review*.

Dahl, Ø., Aune, S. & Berg, T. (2021). Activity-based costing: focus on methodology or understanding the business model - has relevance been lost again? *Journal of cost management*.

Doroudi, S. (2020). The Bias-Variance Tradeoff: How Data Science Can Inform Educational Debates. *AERA Open*, 6 (4). doi: https://doi.org/10.1177/2332858420977208.

GeeksForGeeks. (2022). *Greedy Algorithms*. Tilgjengelig fra: https://www.geeksforgeeks.org/greedy-algorithms/ (lest 14.06.2022).

Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow. Concepts, Tools, and Techniques to Build Intelligent Systems.* 2 utg.: O'REILLY Media, inc.

Glen, S. (u.å). *Mean Squared Error: Definition and Example*. Tilgjengelig fra: https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/mean-squared-error/ (lest 29.07.2022).

Gutta, S. (2020). *Bias and Variance in simple terms!* Tilgjengelig fra: https://medium.com/analytics-vidhya/difference-between-bias-and-variance-in-machine-learning-fec71880c757 (lest 29.07.2022).

Hoff, K. G., Helbæk, M. & Bjørnenak, T. (2017). *Økonomistyring 2. Driftsregnskap og budsjettering*. 6 utg. Oslo: Universitetsforlaget AS.

Huilgol, P. (2019). *Accuracy vs. F1-Score*. Tilgjengelig fra: https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2 (lest 12.07.2022).

IBM. (2021). *What is overfitting?* Tilgjengelig fra: https://www.ibm.com/cloud/learn/overfitting (lest 14.06.2022).

Kim, K.-J. & Han, I. (2003). Application of a hybrid genetic algorithm and neural network approach in activity-based costing. *Expert Systems with Applications*, 24 (1): 73-77. doi: https://doi.org/10.1016/S0957-4174(02)00084-2.

Malekian, A. & Chitsaz, N. (2021). *Advances in Streamflow Forecasting - From Traditional to Modern Approaches*. Chapter 4 - Concepts, procedures, and applications of artificial neural network models in streamflow forecasting: Elsevier.

Mallawaarachchi, V. (2017). *Introduction to Genetic Algorithms — Including Example Code*. Tilgjengelig fra: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3 (lest 14.06.2022).

Minitab. (2014). *How to Interpret a Regression Model with Low R-squared and Low P values*. Tilgjengelig fra: https://blog.minitab.com/en/adventures-in-statistics-2/how-to-interpret-a-regression-model-with-low-r-squared-and-low-p-values (lest 29.07.2022.).

Nichols, J. A., Hsien, W. H. C. & Baker, M. A. B. (2018). Machine learning: applications of artificial intelligence to imaging and diagnosis. *Biophysical reviews,* 11 (1): 111-118. doi: https://doi.org/10.1007/s12551-018-0449-9.

Pedregosa, F. V., G. Gramfort, A. Michel, V., Thirion, B., Grisel, O., Blondel, M. P., P., Weiss, R., Dubourg, V., Vanderplas, J. P., A., Cournapeau, D. B., M. & Perrot, M. D., E. (2011). scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825-2830.

Pupale, R. (2018). *Support Vector Machines(SVM) — An Overview*. Tilgjengelig fra: https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989 (lest 14.06.2022).

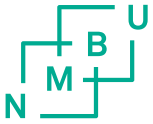
Remeseiro, B. & Bolon-Canedo, V. (2019). A review of feature selection methods in medical applications. *Computers in Biology and Medicine,* 112. doi: https://doi.org/10.1016/j.compbiomed.2019.103375

Sevey, R. (2017). *How Much Data is Needed to Train a (Good) Model?* Tilgjengelig fra: https://www.datarobot.com/blog/how-much-data-is-needed-to-train-a-good-model/#:~:text=For%20example%2C%20if%20you%20have,observations%20to%20train%20a%20model. (lest 12.07.2022).

statisticssolutions. (u. å.). *Pearson's Correlation Coefficient*. Tilgjengelig fra: https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/pearsons-correlation-coefficient/ (lest 12.07.2022).

Toku, A. A. (2021). *Why Feature Scaling in SVM?* Tilgjengelig fra: https://www.baeldung.com/cs/svm-feature-scaling (lest 29.07.2022).

Wickramasinghe, S. (2021). *Bias & Variance in Machine Learning: Concepts & Tutorials*. Tilgjengelig fra: https://www.bmc.com/blogs/bias-variance-machine-learning/ (lest 12.07.2022).



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway