

Optimering av sågverk med maskininlärning



Oliver Bäckström

Division of Industrial Electrical Engineering and Automation
Faculty of Engineering, Lund University

Optimering av sågverk med maskininlärning

Oliver Bäckström

Juni 2025



Masterarbete på

Avdelningen för industriell elektroteknik och automation



LUNDS
UNIVERSITET

Innehåll

1	Introduktion	1
1.1	En automatiserad värld	1
1.2	Beckhoff	1
2	Bakgrund	2
2.1	Ett sågverks anatomi	2
2.2	Processen som studeras	2
2.2.1	Översiktlig processbeskrivning	2
2.2.2	Avståndsmätningen i Dikena	5
2.2.3	Skillnader mellan Elevator 1 och Elevator 2	7
2.2.4	Paketen med plankor	9
2.2.5	Kvalitetens betydelse	9
2.2.6	Tillgänglig information och relevanta optimeringsparametrar	10
3	Problemformulering	12
3.1	Rapportens mål	12
3.2	Frågeställningar	12
3.3	Begränsningar	12
4	Metod	13
4.1	Hands-On Machine Learning	13
4.2	Grundprincipen: Automatisera dataflöden	13
4.3	Databearbetning	13
4.4	Modellval	14
4.5	Optimering av modellparametrar	14
4.6	Utvärdering av modeller	14
4.7	Implementation i sågverkets PLC	15
5	Verktyslådan	16
5.1	Kort introduktion till maskininlärning	16
5.1.1	Varför maskininlärning?	16
5.1.2	Tränings- och testdata	16
5.1.3	Extrapolering	17
5.2	Maskininlärningsmodeller	17
5.2.1	Hur val av maskininlärningsmodell går till	18
5.2.2	Beskrivning av modelltypen Random Forest	19
5.3	scikit-learn och andra relevanta bibliotek	22
6	Implementation	24
6.1	Processkännedom	24
6.1.1	Avgränsningar och antaganden	24
6.1.2	Börja med slutet	25
6.2	Datainsamling	26

6.2.1	Datainsamlingens utveckling	26
6.2.2	Datavariation	28
6.2.3	Kontrollera datainsamlingen	28
6.3	Modellstruktur	29
6.3.1	Vanlig modellstruktur	29
6.3.2	Modellsvårigheter med elevatorsystemet	29
6.3.3	Modellstrukturen för elevatorsystemet	31
6.4	Datafiltrering och datatransformering	31
6.4.1	Borttagning av felaktiga datarader	32
6.4.2	Filtrering	32
6.4.3	Ihopslagning av datafilerna	32
6.4.4	Den momentana maskindatan	34
6.4.5	Datatransformation till paket	34
6.4.6	Labels	35
6.4.7	Filtrering av BPM	35
6.4.8	Filtrering av RDB	36
6.4.9	Tränings- och testset	36
6.5	Val av input och output	36
6.6	Modellval	37
6.7	Automatisk utvärdering av modell	39
6.7.1	BPM	39
6.7.2	RDB	42
6.8	Optimering av maskinparametrarna	47
6.8.1	Optimeringsalgoritmen	47
6.8.2	Parallellisering	47
6.9	Implementation i PLCn	48
6.10	Förbättrad utvärdering av resultatet	49
7	Resultat	52
7.1	Modell och modellstruktur	52
7.2	Modellprestanda	52
8	Diskussion	53
8.1	Varför tillämpa maskininlärning?	53
8.2	Vad kan sägas om resultatet?	53
8.3	Förbättringar i framtiden	53
8.3.1	Predict AND optimize	53
8.3.2	Kvalitet	53
8.3.3	Dynamisk modell	54
8.3.4	Elevator 1	54
8.3.5	Inför en kostnadsfunktion	54
8.3.6	Annan BPM-filtrering i träningsstadiet	55
8.3.7	Ta hänsyn till dåliga parametrar	55
8.3.8	Psykologiska förbättringar	55
9	Slutsatser	56

Figurer

1	Schematisk bild över processen som studeras, sedd från sidan. Plankor kommer från höger och ligger i en rörig klump, men fortfarande med längsriktningen in i figuren. Linjerna i Dikena illustrerar att nivåmätningar utförs där. Streckade pilar visar flödet av plankor. Båda Elevatorer består av rullband som lyfter upp plankor från Dikena. Efter Elevator 2 finns tre olika rullband som var och ett har olika hastigheter. När en plankor når sensorn loggas detta till en databas.	3
2	Paket som ligger och väntar på att köras igenom processen. I bakgrunden ses ett paketlager där plankor i varje fack har samma dimension och kvalitet.	4
3	Inmatning av plankor till Elevator 1. Plankorna har samma dimension och kvalitet. Dikesnivån är högre än för Elevator 2 eftersom syftet här är en första, grövre separation med målet att reglera nivån i Dike 2 mer jämnt och exakt.	4
4	Inmatning av plankor till Elevator 2. Plankor som faller in i Dike 2 kommer från Elevator 1. I jämförelse med Figur 3 är dikesnivån lägre och mer stabilt jämn. Om en dubbel bräda uppkommer under separationsprocessen i Elevator 2 och förs vidare kommer en operatör manuellt behöva separera plankorna i nästa steg. . .	5
5	Separation av plankor inför paketering.	6
6	Paketering av plankor till färdiga paket.	6
7	Distanssensorn.	7
8	Hur Elevator 1 plockar upp plankorna. Det finns klossar som passerar förbi plankorna i Diket som gör att dessa åker upp på bandet.	8
9	Hur Elevator 2 plockar upp plankorna. Till skillnad från Elevator 1 innehåller denna även en justerbar ställskena som anger hur stor greppsmån Elevatorn har vid upplockning av plankor. . . .	8
10	n anger antalet millimetrars greppsmån ställskenan har. Detta räknas från toppen på klossarna och nedåt.	9
11	Två fall av dubbelt plockade bitar vid Elevator 2.	11
12	Dubbel bit som gått vidare till nästa steg i processen.	11
13	Uppdelandet av datan i ett träningsset och ett testset.	17
14	Vanlig linjär regression.	18
15	Träningssetet delas upp i flera mindre delar i korsvalideringssyfte. "CV" står för cross-validation (korsvalidering) och innebär att man väljer ut ett CV-block i taget som får verka såsom valideringsdata. Resterande CV-block används för att träna en modell som sedan valideras mot det utvalda CV-blocket. Detta görs i syfte att finna en modell som är så oberoende från olika datauppdelningar som möjligt.	19

16	Hur korsvalidering (CV) fungerar. I detta fall sker 5 uppdelningar av träningsdata. Därför tränas och valideras 5 iterationer av varje modell för att få information om hur den modellen presterar oberoende av datasetsval. För varje rad väljs det orangea CV-blocket ut till valideringsdata, och en modell tränas på de fyra blå blocken i den aktuella raden. Efter träningen valideras modellen på datan i det orangea blocket. Detta fortgår tills 5 modeller tränats och validerats på de olika datauppdelningarna.	20
17	Ett enkelt beslutsträd bestående av en rotnod, beslutsnoder och slutligen löv som leder till en viss output. Vid en utvärdering finns det två vägar som man kan ta från varje beslutsnod. När ett löv har nåtts returnerar trädet en output motsvarande det som finns i lövet.	20
18	Några enkla exempelträd som kan finnas i en Random Forest-modell. I det verkliga fallet är beslutsträden avsevärt mycket större. En skog kan uppgå till flera hundra eller tusen träd. . . .	21
19	En planka som under processens gång roterats så att den hamnar fel. Detta anses vara en dynamisk effekt och är därför ingenting som modellen tar hänsyn till.	26
20	De två Elevatorerna sedda från sidan. En hög plankor har precis anlänt till Elevator 1 för separation.	27
21	Ett exempel på hur en maskinparameter automatiskt skulle kunna tänkas justeras. I sågverksprocessen används en stegning på 0.5 mm.	28
22	Grafisk beskrivning av den modell som konstrueras i [4] kapitel 2. Huspriserna är understrukna eftersom målet med modellen är att förutspå huspriser så korrekt som möjligt.	30
23	Situationen för modellen i sågverket. Observera att önskan nu är att optimera modellparametrarna, trots att dessa egentligen är <i>inputs</i> till den studerade processen. Modellens möjlighet att förutspå BPM och RDB blir därför ett steg på vägen mot att kunna finna optimala maskininställningar.	30
24	Strukturen för optimeraren som tar hänsyn till att parametrarna som ska optimeras är input till maskininlärningsmodellen. . . .	31
25	Hur den cykliska datan läggs på den diskreta datan förutsatt en backstep på 6 sekunder. Varje rektangel beskriver en loggrad och heltalen representerar loggradens tidstämpel. Pilarna visar att den relevanta momentana maskindatan från de cykliska dataraderna kopieras över till de diskreta dataraderna. I den verkliga processen används en cykeltid för den momentana datan på 250 millisekunder och inte 1 sekund som grafiken antyder på.	33
26	Prestandan av olika baselinemodeller. Inga modellinställningar skruvades på. Varje datapunkt är ett paket. x-axeln beskriver faktisk BPM, y-axeln beskriver modellens förutspådda BPM. Således hade alla datapunkter legat på linjen om modellerna varit perfekta. 38	

27	Resultatet av att ha kört RandomizedSearchCV på vår Random Forest-modell med olika intervall av värden på träd djup och antal träd i skogen. I alla fall har 100 olika parameterkombinationer testats.	40
28	Alternativa intervall för RandomizedSearchCV.	41
29	Slutgiltig korsvalidering av träningsdatan för att finna optimala BPM-modellparametrar. Totalt 100 genomsökningar utfördes. Djup: 15-40, antal träd: 200-400, cv: 5. Bästa: djup 15 med 204 träd.	43
30	Slutgiltig korsvalidering av träningsdatan för att finna optimala RDB-modellparametrar. Totalt 100 genomsökningar utfördes. Djup: 15-35, antal träd: 200-400, cv: 5. Bästa: djup 26 med 254 träd.	45
31	Tillståndsdigram för PLCn	48
32	Filtrering av paket i syfte att sätta ett mer rättvist och oberoende BPM-värde på paketet. Alla x-axlar anger tid. Grafen längst upp ger en bild av hur alla plankor i ett paket passerat sensorn ofiltrerat. Den mellersta grafen visar hur paketet förändras då mellanrum längre än 2 sekunder kortas ner till 2 sekunder. Den sista grafen visar hur paketet förändras då plankkedjor med färre än 10 plankor tas bort. Observera att alla tre grafers x-axlar har olika tidsskalor då hålrum och kedjor med få plankor tas bort. . .	50
33	Samma princip och paket som i Figur 32 men med de grundinställningar som finns på sågverket idag. Observera att antalet hålrum i grafen längst ner här är betydligt fler och större än i Figur 32.	50
34	Samma princip som i Figur 32 men för ett paket där filtreringen inte fungerar bra.	51

Tabeller

1	Parametrar arbetet fokuserar på	10
2	Resultat för olika värden på X för beslutsträdet i Figur 17. . . .	21
3	Modellprestanda för genomsökningarna som finns i Figur 27 och Figur 28. “mean score” anger genomsnittlig avvikelse från korrekt BPM. Denna inkluderar korsvalideringen. Således är lägre värden bättre. “std” är standardavvikelsen från det genomsnittliga inferensresultatet då olika korsvalideringar kördes. “fit time” och “inference time” anger hur lång tid den aktuella modellen tar att träna respektive utvärdera, angivet i sekunder.	41
4	BPM-modellprestanda för de tio bästa resultaten av genomsökningen som finns i Figur 29. “mean score” anger genomsnittlig avvikelse från korrekt BPM. Denna inkluderar korsvalideringen. Således är lägre värden bättre. “std” är standardavvikelsen från det genomsnittliga inferensresultatet då olika korsvalideringar kördes. . . .	44
5	RDB-modellprestanda för de tio bästa resultaten av genomsökningen som finns i Figur 29. “mean score” anger genomsnittlig avvikelse från korrekt RDB. Detta är ett genomsnittligt resultat från alla korsvalideringar. Lägre värden är bättre. “std” är standardavvikelsen från det genomsnittliga inferensresultatet då olika korsvalideringar kördes. De extremt små skillnader som fås när modellinställningarna skruvas på tyder att datan i sig är begränsad och att det inte går att uppnå ett bättre resultat utan vidare filtrering. . . .	46

Begrepp

BPM	Bitar (fysiska brädor/plankor) per minut. Sättet som sågverksmaskinens effektivitet mäts på.
Bräda/Planka	I rapportens syfte används dessa begrepp på samma sätt. Den allmänna distinktionen är att brädor har en mindre tjocklek i förhållande till plankor.
Dike	Den nedsänkning innan varje elevator som matar elevatorn med plankor.
Elevator	En maskin bestående av ett rullband och ställskenor vars syfte är att plocka upp brädor en och en.
KNN	K-nearest neighbors. En maskininlärningsmodell som i detta arbete används i jämförelsesyfte, för att verifiera att den utvalda modellen är speciellt lämplig för processen.
PLC	“Programmable logic controller”, en typ av dator som används inom industrin för att styra maskiner på ett systematiskt sätt.
RDB	Relativ andel dubbla plankor för ett paket plankor.
RF	Random Forest. En vanligen förekommande maskininlärningsmodell.
Ställskenor	Greppskenor som sitter fast längst ner på elevatorerna som ger en viss greppmån för plankorna.

Abstract

As the automation of industrial systems is becoming more and more complex, new methods are needed to further boost efficiency. Machine learning is a method where a computer learns to recognize complex patterns using large amounts of real-world data. The focus of this work is to leverage real-world industrial data from a sawmill to optimize one of the machines, the unscramblers.

The unscrambler separates planks so that they lie next to each other on the track. Sometimes, the unscrambler fails to separate the planks, meaning two planks lie on top of each other. Moreover, the throughput of planks is also important to consider. By tuning the settings of the unscrambler, different results of both efficiency and double boards can be achieved. The goal of the master thesis is ultimately to create a machine learning model which can predict optimal machine parameters for when the process is running in steady state.

In order to design a functioning machine learning model, rigorous studies of the actual process were performed. This affected data collection, filtration and transformation. An evaluation of different machine learning models was performed after which Random Forest was found to be most appropriate. Its model parameters were fine-tuned until a good balance between score and model size was found.

The process exists in a limiting context which means that the machine is sometimes forced to stop through no fault of its own. Because of this, a new efficiency filtration scheme was also developed with the goal of removing dynamic, non-steady-state effects. With the already existing efficiency measuring scheme no visible improvement was detected using the model. Using this newly developed efficiency filtration scheme, it was found that the model performs 5-10% better than the sawmill's reference parameters.

Keywords: machine learning, decision tree, random forest, unscrambler, sawmill, scikit-learn, data collection, data filtration, model evaluation, PLC, industry.

Förord

Ett stort tack till mina ihärdiga handledare Erik Ottosson på LTH och Jon Davidsson på Beckhoff som bidragit med nödvändig kompetens för att detta arbete skulle kunna utföras. När jag har varit vilsen och inte vetat vart jag skulle gå härnäst har jag fått ovärderliga förslag. Själva arbetsmiljön på Beckhoff har varit extremt stöttande och hjälpsam. Tack också till alla operatörer och ansvariga på sågverket som bidragit med så stor villighet att pröva mina idéer.

Jag vill även tacka alla programmerare och utvecklare som gör så kraftfulla verktyg så oerhört lättillgängliga i form av robusta pythonbibliotek. Detta har gjort det mycket lättare att fokusera på kärnan i implementationen istället för att fundera så mycket över de datavetenskapliga aspekterna.

Slutligen vill jag också tacka min fru Isabelle Bäckström och min dotter Miranda Bäckström som genomgående har lyft upp mig och gett mig styrkan att fortsätta med arbetet även när det varit kämpigt. Ni har båda behövt stå ut med en make och pappa som reser iväg och jämt kommer hem trött efter dagens utvecklingsarbete. Jag hade inte kunnat göra detta arbete utan er.

1 Introduktion

1.1 En automatiserad värld

I takt med att världen blir mer komplex och behoven blir fler ökar samtidigt också behovet av att automatisera. Man ska producera mer, öka kvaliteten, öka valbarheten, minska på kostnaderna och göra allting tillgängligt för så många som möjligt.

När de mer uppenbara automatiseringarna har utförts måste man ta till mer avancerade metoder för framtida effektiviseringar. När det inte finns nog med kunskap om en process, eller när processambanden inte är uppenbara behöver tillvägagångssättet förändras. En relativt ny metod som kan lösa dessa problem är maskininlärning.

När vi som människor lär oss nya saker måste vi få se många exempel på hur den sak vi lär oss utförs korrekt. Sedan måste vi få pröva oss fram själva och bli vägledda i rätt riktning. När vi gör fel får vi veta det, och när vi gör något bra gör vi mer av det. Maskininlärning fungerar på motsvarande sätt. Genom att mata in stora mängder data i en maskininlärningsmodell får en sådan modell se många exempel på vad "korrekt" svar är. Om man går igenom denna process tillräckligt länge blir modellen sedan självständig i det att den själv kan producera rimliga svar som bygger på den data den tränats på.

Denna rapport innehåller en industriell tillämpning av maskininlärning. Specifikt är den en del av ett sågverk som optimeras. Med hjälp av maskininlärning förutspås utfallet givet vissa olika maskininställningar. Denna modell är sedan grundbulten i ett optimeringsskal som tar fram optimala maskininställningar.

Även om rapportens huvudsakliga fokus hamnar på den studerade processen är ett delmål att fungera som en översiktlig vägledning för alla som är intresserade av maskininlärningsstillämpningar inom industrin. Genomgående i rapporten extraheras allmänna principer utifrån den praktiska situation som råder i den studerade processen. Detta gör att rapportens disposition är mer flytande och levande i jämförelse med traditionella formella rapporter. Det är ett medvetet val för att underlätta för ingenjörer som önskar tillämpa maskininlärning utan att ha en bakgrund inom området.

1.2 Beckhoff

Beckhoff Automation AB är ett dotterbolag till Beckhoff i Tyskland. Huvudbolaget producerar och levererar produkter och mjukvaror som möjliggör PC-baserad industriell automation. Automationsmjukvaran TwinCAT som är miljön där Beckhoffs PLC-datorer programmeras är mångsidig i det att det finns många moduler med olika tillämpningar såsom maskininlärning, protokolltyper och reglerteknik för att nämna några. Flexibiliteten i lösningen gör att många väljer att vända sig till Beckhoffs produkter för att möta sina automationsbehov.

Själva examensarbetet utförs åt en av Beckhoffs kunder med stöd från Beckhoff.

2 Bakgrund

Detta examensarbete fokus ligger på en specifik process i ett sågverk. Initiiellt ges övergripande information av hur sågverk är uppbyggda för att ge kontext. Sedan beskrivs den process som studeras i detta arbete i större detalj.

2.1 Ett sågverks anatomi

I ett sågverk sker mängder av processer för att stockar ska omvandlas till plankor som finns att köpa som konsument. Först anländer stockar av olika storlekar som sorteras efter dimension. När man sedan påbörjar en körning väljer man ut en specifik storlek av stockar och håller sig till dem.

Stockarna förs in i ett kontinuerligt flöde av maskiner som var och en processar stockarna. Barken avlägsnas och stocken sågas till plankor. Brädorna plockas sedan ut en i taget. Individuellt analyseras de utifrån dimension och förs sedan vidare till ett fack motsvarande aktuell dimension.

När ett sådant sorteringsfack blivit fullt töms plankorna och de separeras återigen. Plankorna paketeras med separationsplankor mellan lagren. Detta görs inför torkningen av virket för att även plankor inuti paketet ska kunna torka.

Paketen med plankor skickas sedan vidare till sågverkets torkanordning för att minska fuktandelen i virket. När de har blivit tillräckligt torra går de vidare till nästa del där dessa återigen separeras så att en plankor i taget plockas ut. Därefter analyseras varje individuell plankor för att avgöra hur denna ska kapas för att producera mest värde. Plankan sorteras efter dimension och kvalitet och slutpaketeras sedan för att skickas ut till återförsäljaren.

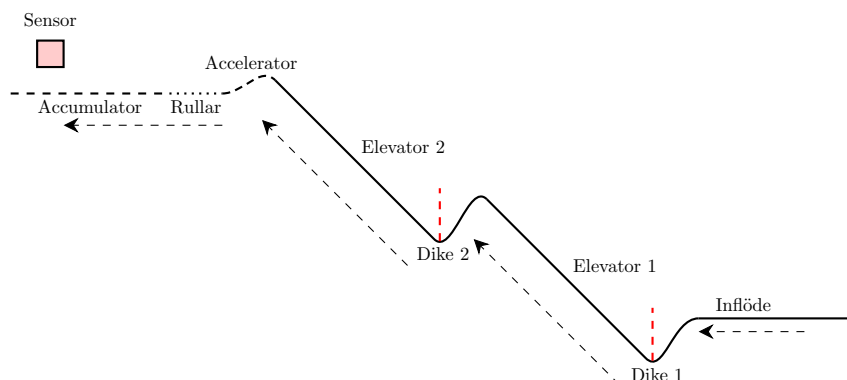
Var och en av dessa delmoment kan man spendera åtskilliga examensarbeten på. Detta torde däremot ge den relevanta översiktliga förståelsen av sågverk som krävs för att förstå kontexten till det aktuella arbetet.

2.2 Processen som studeras

2.2.1 Översiktlig processbeskrivning

Detta examensarbete berör den del av sågverksprocessen som sker efter torkning av plankorna. När dessa torkats färdigt görs en bedömning av hur man kapas plankorna bäst för att få så hög lönsamhet som möjligt. Plankorna sorteras sedan i fack efter dimension och kvalitet. Därefter släpper man plankorna från ett fack i taget. De ska sedan plockas upp en och en för att dessa ska kunna paketeras effektivt. Denna separation av plankorna en och en är den process som ska optimeras under arbetets gång. En systematisk beskrivning av de maskiner som bedriver plankseparationen finns i Figur 1.

I figuren betraktas ett tvärsnitt av processen och plankornas längsriktning går in i figuren. Det första stoppet plankorna når är Dike 1. Här finns en avståndsmätare som mäter djupet i Diket. Med hjälp av detta går det att få en översiktlig bild av hur många plankor som finns i Diket vid varje stund. I maskinen finns en automatisk reglering av denna dikesnivå. Det finns ett börvärde



Figur 1: Schematisk bild över processen som studeras, sedd från sidan. Plankor kommer från höger och ligger i en rörig klump, men fortfarande med längsriktningen in i figuren. Linjerna i Dikena illustrerar att nivåmätningar utförs där. Streckade pilar visar flödet av plankor. Båda Elevatorer består av rullband som lyfter upp plankor från Dikena. Efter Elevator 2 finns tre olika rullband som var och ett har olika hastigheter. När en plankor når sensorn loggas detta till en databas.

för vad nivån ska sträva efter och hastigheten på inflödesbandet justeras utifrån detta. Kortfattat kan man säga att inflödesbandets hastighet är proportionell mot dikesnivåfelet, då Diket har ett underskott av plankor. När det finns ett överskott av plankor i ett Dike stannar Dikets matarband helt. Det är först när felet blir stort nog som bandet börjar mata på nytt.

På grund av plankornas diskreta natur och det faktum att de i detta skede ligger i en hög är det svårt att få en kontinuitet i flödet. Ibland kan en matning av inflödesbandet med totalt 20cm resultera i en stor hög med plankor i Dike 1, medan i en annan situation kan en matning med 50cm resultera i en liten hög i Dike 1. Egentligen sker inmatningen genom att bandets hastighet kontrolleras. Matningarna anges i cm för att illustrera att matningen är olinjär i sin natur. Figur 2 visar inflödet av plankor till systemet och Figur 3a visar inflödet av plankor till Dike 1.

Elevator 1 plockar kontinuerligt upp plankor från Dike 1. På Elevatorn finns utbuktande klossar som gör att dessa tas upp. Idealt tas plankorna redan i detta skede upp en och en. Syftet med den första Elevatorn är att göra en första, grövre separation av plankorna. Genom att ha en sådan första buffer är det möjligt att kontrollera nivån i Dike 2 mer exakt, vilket leder till bättre prestanda för Elevator 2. Hastigheten för Elevator 1 beror på nivån i Dike 2, på motsvarande sätt som inflödesbandets hastighet beror på nivån i Dike 1. Bilder på Elevator 1 finns i Figur 3.

Det stora fokuset av arbetet ligger på Elevator 2. I Figur 4 finns bilder från den faktiska processen. Elevator 1 matar Dike 2 och ser till så att nivån ligger så nära börnivån som möjligt. Även här sker en avståndsmätning i detta



Figur 2: Paket som ligger och väntar på att köras igenom processen. I bakgrunden ses ett paketlager där plankor i varje fack har samma dimension och kvalitet.



(a) Ovan.



(b) Framifrån.

Figur 3: Inmatning av plankor till Elevator 1. Plankorna har samma dimension och kvalitet. Dikesnivån är högre än för Elevator 2 eftersom syftet här är en första, grövre separation med målet att reglera nivån i Dike 2 mer jämnt och exakt.

syfte. Det är prestationen av Elevator 2 som blir det ytterst viktiga resultatet av processen. Efter Elevator 2 ska plankorna vara separerade då de kommande maskinerna i sågverksprocessen kräver detta. För att främja detta finns det ett antal maskinparametrar som var och en bidrar till resultatet på ett eller annat sätt. Dessa kommer beskrivas i detalj senare.



(a) Medioker plockning.



(b) Valfylld plockning.

Figur 4: Inmatning av plankor till Elevator 2. Plankor som faller in i Dike 2 kommer från Elevator 1. I jämförelse med Figur 3 är dikesnivån lägre och mer stabilt jämn. Om en dubbel bräda uppkommer under separationsprocessen i Elevator 2 och förs vidare kommer en operatör manuellt behöva separera plankorna i nästa steg.

Elevator 2 plockar upp plankor från Dike 2 kontinuerligt så länge flödet framåt i processen inte gör det nödvändigt att pausa bandet. När en plankor väl kommit över Elevator 2 når den ett accelerationsband som ytterligare bidrar till att plankorna separeras ifrån varandra. Rullarna ser till så att plankorna ligger på samma plats i sin längsriktning. De flyttas alltså inte bara framåt, utan också i längsriktningen. Accumulatorn tar sedan plankorna till nästa delsystem. Här sitter också den sensor som triggar loggningen av att en plankor har passerat. Här finns också automatiska beräkningar som tar reda på om maskinen framgångsrikt lyckats separera plankan eller inte. Dessa beräkningar baseras på plankornas tjocklek. Om sensorn upptäcker att det under sensorn befinner sig material motsvarande två gånger en plankas tjocklek loggas den aktuella datapunkten som en dubbel plankor.

När plankorna väl har separerats en och en går de ut mot paketering. I Figur 5 går det att urskilja hur en separatormaskin ser till så att rätt antal plankor läggs på varje lager i paketet. Därefter går dessa grupperingar vidare till att paketeras av paketläggaren i Figur 6.

2.2.2 Avståndsmätningen i Dikena

Eftersom ett sågverk hanterar diskreta produkter finns det många dynamiska effekter som försvårar nivåmätningarna. Om en plankor står på bredden kommer nivån i Diket att betraktas som högre än vad det egentligen är. Plankornas natur gör också att denna nivå skiftar snabbt och sporadiskt och kräver att kraftig filtrering utförs på denna. Figur 7 visar nivåmätningen i Dikena.



Figur 5: Separation av plankor inför paketering.



Figur 6: Paketering av plankor till färdiga paket.



Figur 7: Distanssensorn.

2.2.3 Skillnader mellan Elevator 1 och Elevator 2

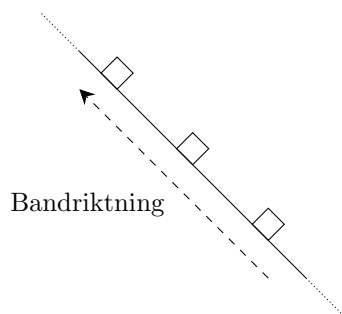
För båda Elevatorer kan man sätta önskad dikesnivå i Dike 1 och Dike 2. En högre nivå innebär att plankorna utsätts för större tryck vilket ökar sannolikheten att Elevatorerna plockar upp plankor. Detta ökar däremot också sannolikheten för att en dubbel plankor plockas upp. Det bör därför finnas någon typ av kompromiss som ger det bästa av båda världar.

Eftersom resultatet efter Elevator 2 är av större vikt än Elevator 1 finns det flera extra parametervärden som kan sättas där. För det första så kan man sätta bandets maximala och minimala hastighet. Anledningen till att parametervärdena kontrollerar min/maxhastigheterna är för att den faktiska hastigheten baseras på hur stor möjlighet det finns att mata in plankor i den kommande processen. I steady-state så körs Elevator 2 med maxhastigheten. Anledningen till att det finns en minihastighet är för att en hastighet som är för låg är inte önskvärd i processen. Vid lägre hastigheter ökar sannolikheten för dubbla brädor avsevärt. Om den interna börshastigheten för Elevator 2 skulle sjunka under minihastigheten stannar bandet tills börvärdet går upp igen.

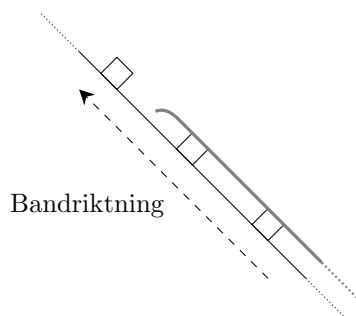
I Figur 8 ses en grafisk beskrivning av Elevator 1 sedd från sidan. Det som gör att plankor plockas upp är det faktum att det sitter fast klossar på bandet. När klossarna passerar Diket fastnar brädor på klossarna och gör så att dessa färdas upp på bandet. Då syftet med Elevator 1 är att göra en grov första sortering duger detta tillvägagångssätt.

I Elevator 2 är situationen något förändrad. Eftersom det nu är av största vikt att minimera andelen dubbla plankor finns det möjlighet att i Elevator 2 kontrollera greppmån för bandet. I Figur 9 går det att se att situationen liknar den för Elevator 1, fast med en så kallad ställskena tillagd. Denna ställskena kan justeras för att på millimeternivå kontrollera bandets greppmån. En närmare titt på hur detta ser ut finns i Figur 10.

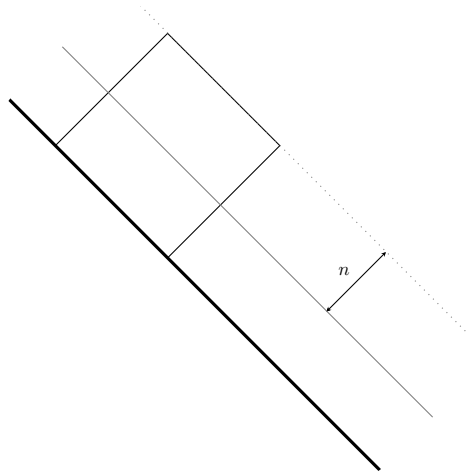
Inbyggt i ställskenorna finns också en viss hysteres. Det innebär i praktiken att ställskenan inte kommer att justera om sig vid en börvärdesförändring om börvärdesförändringen inte är så stor. Syftet med detta är att förhindra



Figur 8: Hur Elevator 1 plockar upp plankorna. Det finns klossar som passerar förbi plankorna i Diket som gör att dessa åker upp på bandet.



Figur 9: Hur Elevator 2 plockar upp plankorna. Till skillnad från Elevator 1 innehåller denna även en justerbar ställskena som anger hur stor greppsmån Elevatorn har vid upplockning av plankor.



Figur 10: n anger antalet millimetrars greppsmån ställskenan har. Detta räknas från toppen på klossarna och nedåt.

självsvängningar. Nackdelen blir att kontrollen av ställskenorna inte är lika precis. Detta skulle kunna lösas med mer noggrann styrutrustning men i detta arbete är det inget som kan förändras.

2.2.4 Paketen med plankor

Varje gång ett nytt paket med plankor presenteras för elevatorsystemet läses information in om de plankorna. Information om bredd, tjocklek, längd, plankkvalitet och batchnummer läses in. Värdena ligger sedan som grund till de basinställningar som sätts för Elevatorerna. Dessa “recept” har tagits fram experimentellt och har visat sig fungera tillräckligt bra. Det finns däremot säkerligen mer optimala maskininställningar som man har missat genom att nöja sig med de inställningar man kör med idag.

Varje paket med dessa plankor är tydligt separerade och oberoende från varandra. Det är inte förrän ett givet paket helt har passerat elevatorsystemet som nästa paket lastas in.

2.2.5 Kvalitetens betydelse

Varje paket med plankor har en kvalitet associerad till sig. Denna kvalitetsbedömning görs automatiskt genom lasersensorer. Brädornas skevhet, rätthet och kontur mäts, samt detekteras eventuella sprickor i virket. Plankor av lägre kvalitet kan ofta vara svårare att plocka upp av Elevatorerna.

2.2.6 Tillgänglig information och relevanta optimeringsparametrar

Det finns en hel del information från denna process som går att läsa av från systemets sensorer. Följande är en översiktlig beskrivning av dessa som följer flödet av plankor som finns i Figur 1.

- Börvärde och faktiskt värde på dikesnivån för båda Dikena
- Faktisk hastighet för båda Elevatorerna
- Min- och maxhastighet för Elevator 2
- Börvärde och faktiskt värde för ställskenor till Elevator 2
- Faktiska bandhastigheter på de tre olika banden som finns efter Elevator 2

I detta arbete kommer fokus vara att optimera de parametrar som troligtvis har störst påverkan på systemets effektivitet och robusthet enligt sågverkspersonalens intryck. Dessa finns i Tabell 1.

Parameter	Beskrivning
<code>min_speed</code>	Den minimalt tillåtna hastigheten för Elevator 2.
<code>max_speed</code>	Den maximalt tillåtna hastigheten för Elevator 2.
<code>level</code>	Börvärdet på nivån för Dike 2 som matar Elevator 2 med plankor.
<code>rails</code>	Börvärdet för de ställskenor som anger vilken greppmån Elevator 2 har i upplöckningen av plankor.

Tabell 1: Parametrar arbetet fokuserar på

Viktigt att notera är att dessa parametrar i sin tur bildar ramverket för maskinens befintliga regulatorer.

För att mäta hur effektivt systemet fungerar finns det två nyckeltal som tillsammans bildar en rättvis bild av processens prestation. Dessa är BPM och RDB. BPM står för bitar per minut och anger hur många plankor som passerat sensorn den senaste minuten. RDB innebär "relativ andel dubbla brädor" för ett paket brädor.

I processen sker många naturliga stopp som inte kan anses vara maskinens underprestation. För att därför ge en så rättvis bild som möjligt pausas samplingen av brädor om ingen ny bräda detekterats på 5.5 sekunder. Detta gör att BPM-värdet inte går mot 0 då maskinen pausats ett tag. Om detta inte hade gjorts så hade maskinens prestandamätning varit extremt beroende av externa faktorer och hade därför inte sagt så mycket om maskinens faktiska prestation.

Ett mätvärde klassas som en så kallad "dubbel bräda" om två brädor ligger staplade på varandra. I Figur 11 finns två exempel på när dubbla brädor uppkommit då Elevator 2 plockat plankor. I Figur 12 går det att se en dubbel plankor som gått vidare till nästa steg. Här måste en operatör manuellt vända plankan för att maskinen inte ska fastna i ett senare skede.

I det ideala fallet uppkommer inga dubbla brädor samtidigt som BPM hålls konstant hög.



(a) Långt upp på bandet. En böjd plank har lagt sig ovanpå en annan.



(b) Mitt på bandet, på väg upp.

Figur 11: Två fall av dubbelt plockade bitar vid Elevator 2.



Figur 12: Dubbel bit som gått vidare till nästa steg i processen.

3 Problemformulering

3.1 Rapportens mål

Hela syftet med detta examensarbete är att visa på en praktisk implementering av maskininlärning inom industrin i optimeringssyfte. Även om fokus ligger på själva sågverksprocessen extraheras också principer som är användbara oavsett typ av process. Alla inom industrin som på något sätt önskar få en inblick i hur maskininlärning kan tillämpas rent praktiskt ska kunna hämta något från denna rapport.

3.2 Frågeställningar

- Vilka maskininlärningsmodeller lämpar sig för parameteroptimering av en industriell maskin?
- Vilken modellstruktur lämpar sig där datan inte innehåller korrekt inställda parametrar?
- Hur ser implementationsprocessen av maskininlärningsmodellen ut i praktik?
- Kan maskininlärning förbättra parameterinterval av sågverksprocessen trots att datan innehåller inoptimalt inställda maskinparametrar?
- I hur stor utsträckning leder maskininlärningsmodellen till ökad BPM och reducerat antal dubbla brädor?
- Tillför maskininlärningsmodellen något som traditionell analys inte hade kunnat bidra med?

3.3 Begränsningar

I processen sker många dynamiska effekter som tvunget kräver en operatörs handpåläggning för att lösa. Detta är inget som modellen i detta arbete ska ta hänsyn till. Istället blir arbetets fokus att optimera maskinparametrar utifrån när maskinen körs i kontinuerlig drift.

Egentligen finns fler parametrar än de som anges i Tabell 1. Till exempel går det även att justera nivån i Dike 1. Eftersom det är resultatet efter Elevator 2 som spelar störst roll i optimeringen ligger därför fokus på att optimera denna.

I övrigt beaktas maskinen i sitt befintliga skick. Inga maskinella förändringar får ske förutom i datainsamlingssyfte.

4 Metod

I detta kapitel kommer de metoder och teorier som använts under examensarbetets implementationsdel att översiktligt förklaras.

4.1 Hands-On Machine Learning

Den allmänna implementationsmetodiken av maskininlärningsmodeller tas ifrån boken “Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow”. [4] Denna bok beskriver vad som behöver beaktas när man arbetar med maskininläring, vanliga fallgropar och den ger också exempel på hur maskininläringssystem implementeras i praktiken.

Boken ger däremot enbart allmänna principer och har inte industrin som sin primära målgrupp. Boken kommer därför att användas som utgångspunkt och detta examensarbete ger ett tillvägagångssätt mer specialiserad mot industrin. Inom datavetenskapliga tillämpningar finns det ofta stora datamängder tillgängliga. Industriella tillämpningar består alltid av fysiska maskiner och måste interageras med hjälp av sensorer. Sensorer är bristfälliga och datan som kommer ifrån dessa måste bearbetas. I övrigt är det också relevant hur man väljer att kvantifiera den fysiska processen. Var ska sensorer sättas upp? Vilken data ska samlas in? Svaret på dessa frågor har en stor inverkan på resten av modellkonstruktionen.

4.2 Grundprincipen: Automatisera dataflöden

Inom maskininläring arbetar man med stora mängder olinjär data. Inom en industriell kontext sker dessutom datainsamlingen ofta kontinuerligt. Det betyder att den datamängd man arbetar med inte är statisk utan kan komma att förändras. För att ta hänsyn till den ständigt föränderliga datamängden bör alla maskininlärningsprojekt mynna ut i automatiserade dataflöden.

För att ta reda på vilka typer av flöden som ska implementeras måste engångstransformationer utföras på datan. När önskvärd algoritm identifierats ska istället automatiserade transformationer implementeras. I detta syfte används pythonklassen `Pipeline` som finns i maskininlärningsbiblioteket `scikit-learn`. [12] En kort genomgång av hur detta fungerar finns beskrivet i 5.3.

Den stora fördelen med att arbeta på detta sätt är att man slipper väldigt mycket dubbelarbete. Allt som krävs för att en ny datamängd ska bli redo att köras igenom en modell är en knapptryckning.

4.3 Databearbetning

Industriell data innehåller i regel mycket brus. Sensorer meddelar felaktiga värden, datapunkter saknas och så vidare. Maskininlärningsmodellen ser datan vi matar in som absolut sanning. Det man brukar säga om data gäller i högsta grad också här:

Garbage in, garbage out.

I detta syfte ska datans brister undersökas. Vilka värden behöver filtreras, och hur ska de filtreras? Hur ska felaktiga datarader upptäckas och vad ska göras åt dem?

Det räcker inte att enbart korrigera datans brister. Datan behöver också omvandlas och bearbetas för att vara mer användbar för en maskininlärningsmodell. Kan datan transformeras på något sätt för att öka modellens effektivitet? Finns det tidsförskjutningar som behöver beaktas? Vilken detaljrikedom behövs vid detta moment?

4.4 Modellval

Det finns en uppsjö av olika maskininlärningsmodeller. Principen bakom dem alla är liknande: få en dator att lära sig se samband av sig själv med hjälp av faktisk data. Sättet som de är utformade på gör däremot att vissa är mer lämpade än andra för olika syften.

Eftersom hela syftet med maskininlärning är att få en modell att lära sig av sig själv, är maskininlärning som vetenskap synnerligen experimentell. [4, 7, 13] Därför kommer val av optimal maskininlärningsmodell också att ske experimentellt. Om tillräckligt med kunskap över datan funnits för att kunna säga vilken modell som hade lämpat sig bäst, hade man säkerligen istället kunnat tillämpa enklare, mer traditionella analytiska metoder. Tillvägagångssättet blir således att träna flera olika modeller på samma mängd data och analysera vilken modell som är bäst.

4.5 Optimering av modellparametrar

Varje typ av maskininlärningsmodell har olika inställningar som kan varieras för att förändra (förhoppningsvis till det bättre) modellens prestanda. När en modell väl har valts kommer samma modell fast med olika inställningar att tränas om flera gånger. Genom att göra detta går det att se vilken inställningskombination som gav bäst resultat. I `scikit-learn` finns det bibliotek som stödjer automatisk utvärdering av olika inställningar. I denna rapport kommer `RandomizedSearchCV` att användas, vilket är en slumpad genomsökning av olika maskininställningar.

4.6 Utvärdering av modeller

Inom maskininlärning finns det ofta tillgång till korrekt data i förhållande till vad vi vill att modellen ska ge som output. Om man skapar en maskininlärningsmodell som ska förutspå vilken blomma en växt är genom färg, utseende, längd etc. kräver detta att det finns data som innehåller alla inputvärden men också vilken *typ* av blomma som data beskriver. Om man ger modellen tillräckligt många med sådana exempel ökar sannolikheten för att den lyckas klassificera blommorna korrekt.

I vårt fall vill vi optimera *maskininställningarna*. I denna bemärkelse går det inte att säga att datan är korrekt. Vi vet inte vilka maskininställningar som är mest optimala. Det är modellens syfte att ta reda på detta åt oss. Men om datan inte är korrekt utifrån perspektivet av det som ska optimeras, hur kan vi gå tillväga? Metodiken av hur detta löses blir att titta på hur maskininlärning tillämpats tidigare i industriella kontexter. [8, 15]

4.7 Implementation i sågverkets PLC

Själva praktiska implementationen ska göras i sågverkets Beckhoff-PLC. Själva maskininlärningsmodellen ska implementeras i Python. Kommunikation mellan PLC-programmet och Pythonprogrammet åstadkomms genom att PLCn anropar Pythonprogrammet från kommandotolken och läser dess resultat från en textfil.

5 Verktygslådan

I denna del ges en kortfattad introduktion till maskininlärning för de som inte har stött på ämnet tidigare. Tanken är inte att vara heltäckande, utan att ge den bakgrund som krävs för att förstå det aktuella examensarbetet. För vidare studier inom maskininlärning rekommenderas [4]. Om man redan har en förståelse för grundläggande maskininlärningskoncept kan man hoppa över denna del.

5.1 Kort introduktion till maskininlärning

All maskininlärningsrelaterad information i denna del är tagen från [4] om inget annat anges.

5.1.1 Varför maskininlärning?

När man har tillgång till stora mängder data men det inte ser ut att finnas några enkla samband att extrahera från den datan kan maskininlärning lämpa sig för att finna dessa mönster. Istället för att manuellt lägga ner mycket tid och energi på att analytiskt organisera datan kan man låta en dator finna smarta sätt att utföra den organiseringen på. En maskininlärningsmodell kan också enkelt tränas om på ny data, vilket gör att samma metodik också kan fungera vid förändringar i datan.

Grundprincipen inom maskininlärning är att designa ett system som i någon mening “lär sig själv”. Genom att sedan ge modellen tillgång till mycket data i form av input och önskvärd output, kan modellen ställas in (tränas) så att den motsvarar datan. Med lite skicklighet kan denna modell sedan köras på ny data som den aldrig tidigare sett och förutspå önskat utfall med stor sannolikhet.

5.1.2 Tränings- och testdata

Eftersom allting inom maskininlärning utförs experimentellt är det nödvändigt att resultaten går att verifiera. Om man använder all sin data till att träna en modell, hur kan man då vara säker på att modellens prestanda också går att generalisera till data modellen inte har sett? Visst skulle det gå att köra träningsdatan genom modellen och se hur bra modellen lyckas utvärdera den. Problemet är att eftersom modellen har tränats på denna data bör modellen prestera extra bra på denna. Om man vill vara säker på att modellen faktiskt generaliseras bra är det nödvändigt att testa modellen på ny data. I detta syfte delar man upp hela sitt dataset i två delar. Ena delen är det så kallade träningssetet, och den andra delen är testsetet. Testsetet läggs åt sidan. Detta ska inte beaktas under processens gång. Det är enbart senare i projektet innan modellen tas i drift som denna används till att verifiera att modellen faktiskt är användbar. Under hela processens gång är det alltså testdatan som bearbetas. En vanlig uppdelning är att 80% av datan blir träningsset och 20% blir testset. Detta illustreras i Figur 13

All data	
Träningsdata	Testdata

Figur 13: Uppdelandet av datan i ett träningsset och ett testset.

När denna uppdelning görs är det fördelaktigt att basera urvalet på något statistiskt i datan, exempelvis varje loggrads ID, så att uppdelningen görs deterministisk. Annars kommer olika tränings- och testset att skapas vid olika körningar av programmet. Det kan göra det svårare att se om förändringar som görs faktiskt leder till en förbättring av modellen eller om förändringen kom ur det faktum att träningssetet nu ser annorlunda ut. Det finns färdiga metoder för att göra uppdelningen av dataset deterministiska, men detta förutsätter att datasetet inte förändras. Med andra ord kan det räcka ett en datarad läggs till för att uppdelningen i tränings- och testset ska vara helt annorlunda. Genom att grunda uppdelningen i något som inte förändras även när datamängden ökar ser man till så att jämförelsen blir mer stabil.

5.1.3 Extrapolering

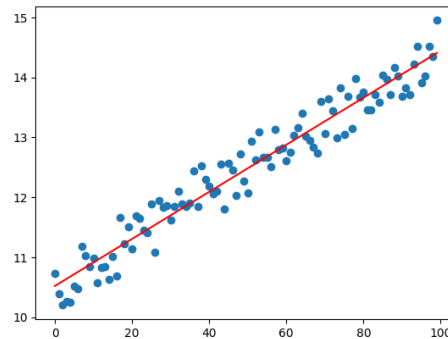
Om en tränad modell ska utvärderas på en datapunkt som modellen inte har någon tydlig data för tvingas modellen att extrapolera. Baserat på datan gör modellen en kvalificerad gissning. Det går såklart inte att garantera att en sådan gissning stämmer överens med verkligheten, i synnerhet när systemet som modelleras är olinjärt. Detta är varför bra spridning på datan är nödvändig.

5.2 Maskininlärningsmodeller

En maskininlärningsmodell kan upplevas mystisk och komplicerad. Det finns en bild om att dessa är svåra att förstå och förklara och mycket göms bakom termer såsom "AI". Det stämmer visserligen att det inte alltid går att säga *varför* en modell beter sig som den gör. Men principerna bakom hur de olika modellerna är konstruerade är väldigt handfasta och logiska.

I grund och botten är syftet bakom modellen att beskriva verkligheten på något sätt, och att göra det så bra som möjligt. Ett beskrivande exempel som alla ingenjörer stött på vid en eller annan tidpunkt är linjär regression. I någon mening kan man säga att algoritmen bakom denna är en form av enkel maskininlärning. Det handlar om att skapa en linje som så nära som möjligt kan beskriva data. Ett exempel är den linje som skapats åt datapunkterna i Figur 14.

Maskininlärningsmodeller kan man se som avancerade versioner av den enkla linjära regressionen. Dessa modeller observerar indatan (de datapunkter som finns), och justerar sedan sin interna konstruktion (tränas) för att så nära som möjligt beskriva utdatan med utgångspunkt ur indatan. På samma sätt som



Figur 14: Vanlig linjär regression.

i Figur 14 kan vi, när modellen väl är på plats, förutspå vilket y-värde som ett givet x-värde rimligtvis leder till. Istället för att enbart ha *en* input och *en* output går det att ha en godtycklig mängd in- och outputs.

5.2.1 Hur val av maskininlärningsmodell går till

Som tidigare har nämnts är maskininläring experimentellt, vilket innebär att även val av modell utförs med hjälp av experiment. Det finns modeller som presterar bra i de flesta fallen, men det är ingen garanti att de gör det för den aktuella tillämpningen. För att ta reda på vilken modell som lämpar sig bäst bör flera modeller tränas på datan innan den bästa modellen väljs ut för vidare studier.

När man testar hur väl en modell presterar räcker det inte att nöja sig med att enbart träna en modell och utvärdera den på enbart en uppdelning av tränings- och testdatan. För att kontrollera att modellen konsekvent generaliseras bra måste flera, olika uppdelningar av datan utföras. För varje uppdelning av datan ska en modell tränas på ena datadelen och sedan valideras mot den andra. I och med att modellen med samma parametrar tränas och testas på olika delar av datan fås en så rättvis bild som möjligt av hur bra modellen skulle kunna generaliseras. Som tidigare har nämnts ska inte testsetet användas förrän precis innan slutimplementation, vilket är varför dessa uppdelningar görs av träningssetet. Konceptet illustreras i Figur 15. I detta fall används 5 uppdelningar.

Efter det att träningsdatan delats upp är det dags att utföra valideringen. Detta görs genom att ett set i taget väljs ut som valideringsset. Modellen tränas på resterande data och valideras på valideringssetet. Det betyder att lika många modeller ska tränas som datauppdelningar görs. Själva valideringsprocessen förtydligas i Figur 16. Genom att sedan ta ett genomsnitt av modellens prestation över alla olika uppdelningar fås ett resultat som är mer robust. Risken är annars att en modell hade passat väldigt bra för ett specifikt tränings- och testset, men inte hade generaliserats bra alls.

All data					
Träningsdata					Testdata
CV 1	CV 2	CV 3	CV 4	CV 5	

Figur 15: Träningssetet delas upp i flera mindre delar i korsvalideringssyfte. “CV” står för cross-validation (korsvalidering) och innebär att man väljer ut ett CV-block i taget som får verka såsom valideringsdata. Resterande CV-block används för att träna en modell som sedan valideras mot det utvalda CV-blocket. Detta görs i syfte att finna en modell som är så oberoende från olika datauppdelningar som möjligt.

Trots att korsvalidering utförs skyddas man inte helt från att modellen övertränas. Om man tillåter för höga modellvärden kan man fortfarande få problemet att modellen enbart beskriver det aktuella datasetet väl och inte kan generaliseras. Korsvalidering hjälper till för att förhindra detta men är ingen garanti.

5.2.2 Beskrivning av modelltypen Random Forest

Det finns många olika typer av maskininlärningsmodeller. Eftersom detta enbart är en kortfattad inblick i hur maskininläring fungerar beskrivs enbart den modell som slutligen användes i arbetet. För den intresserade läsaren går det att läsa på om olika modelltyper i dokumentationen för `scikit-learn`.

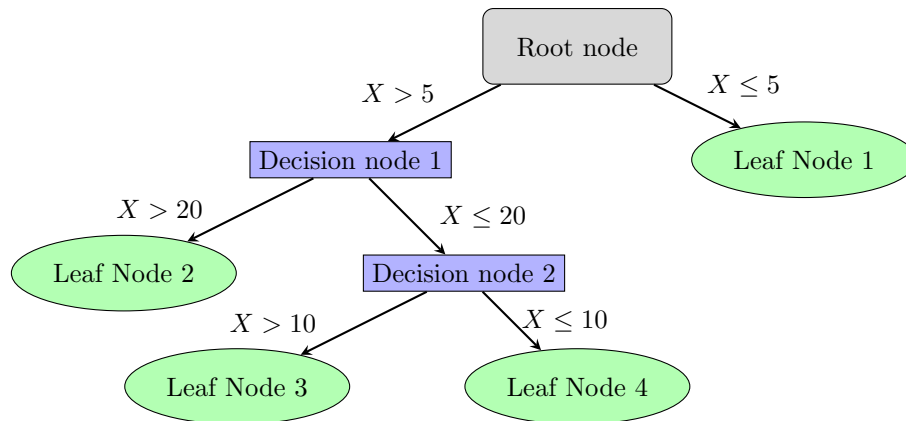
En vanligt förekommande maskininlärningsmodell är Random Forest, eller RF som den ofta förkortas. Modellen är en så kallad *ensemble*-modell, vilket bygger på att modellen består av många mindre modeller som är sammankopplade. Grunden till RF är beslutsträd. Först ges därför en kortfattad beskrivning av hur dessa fungerar.

Ett enkelt beslutsträd illustreras av Figur 17. Varje beslutsträd består av noder. Det finns en rotnod samt beslutsnoder och “löv”. Varje beslutsnod har endast två avgränsningar. Dessa är på ett eller annat sätt ihopkopplade med pilar. Varje pil har ett logiskt påstående knutet till sig. När en utvärdering av ett beslutsträd påbörjas börjar man i rotnoden. Utifrån given input färdas man sedan igenom trädet tills man har navigerat till ett löv. Det lövet producerar därefter någon form av output vilket returneras. Utvärdering av beslutsträdet i Figur 17 finns i Tabell 2.

När det nu har konstaterats att beslutsträd kallas just för *träd*, kan man börja ana var modellnamnet Random Forest kommer ifrån. Det är för att denna modell består av en hel skog av beslutsträd. När dessa träd tränas görs de medvetet medelmåttiga. När man utvärderar en RF-modell utvärderas varje individuell träd och resultatet blir ett genomsnitt av det svar alla kom med. Genom att

Träningsdata					Testdata
CV 1	CV 2	CV 3	CV 4	CV 5	
CV 1	CV 2	CV 3	CV 4	CV 5	
CV 1	CV 2	CV 3	CV 4	CV 5	
CV 1	CV 2	CV 3	CV 4	CV 5	
CV 1	CV 2	CV 3	CV 4	CV 5	

Figur 16: Hur korsvalidering (CV) fungerar. I detta fall sker 5 uppdelningar av träningsdata. Därför tränas och valideras 5 iterationer av varje modell för att få information om hur den modellen presterar oberoende av datasetsval. För varje rad väljs det orangea CV-blocket ut till valideringsdata, och en modell tränas på de fyra blå blocken i den aktuella raden. Efter träningen valideras modellen på datan i det orangea blocket. Detta fortgår tills 5 modeller tränats och validerats på de olika datauppdelningarna.

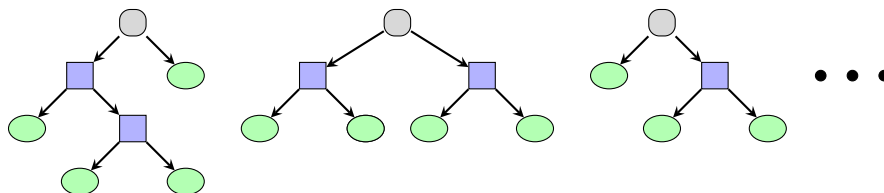


Figur 17: Ett enkelt beslutsträd bestående av en rotnod, beslutsnoder och slutligen löv som leder till en viss output. Vid en utvärdering finns det två vägar som man kan ta från varje beslutsnod. När ett löv har nåtts returnerar trädet en output motsvarande det som finns i lövet.

Input	Result
$X \leq 5$	Leaf Node 1
$5 < X \leq 10$	Leaf Node 4
$10 < X \leq 20$	Leaf Node 3
$20 < X$	Leaf Node 2

Tabell 2: Resultat för olika värden på X för beslutsträdet i Figur 17.

låta resultatet från den färdiga modellen vara ett genomsnittligt värde tar de individuella trädens lokala fel ut varandra. En mer verklighetstrogen prediktion fås. I Figur 18 kan man se principen bakom detta.



Figur 18: Några enkla exempelträd som kan finnas i en Random Forest-modell. I det verkliga fallet är beslutsträden avsevärt mycket större. En skog kan uppgå till flera hundra eller tusen träd.

Hela detta koncept bygger på att det finns styrka i att ta in resultatet av flera prediktioner och samla dessa till en. Man kan se detta som en datavetenskaplig tillämpning av så kallad “crowd forecasting”. Termen innebär att låta en större mängd människor, oavsett bakgrund, väga in i olika prediktionsfrågor. Det går då att se att felaktiga prediktioner som utförs på individuell nivå i regel tar ut varandra och att prediktionen i det stora hela blir säkrare. [10] Ett nutida exempel på hur detta används är en satsning av Totalförsvarets forskningsinstitut vid namn *Glimt*. Syftet bakom detta är att ge Ukraina tillgång till tillförlitliga prediktioner om vad som ska hända i världsläget. För att göra dessa prediktioner låter man allmänheten svara vad man tror att vissa konkreta händelser mynnar ut i för ofall. Det har visats att detta ger robusta prediktioner. [3] Trots att varje individ har en kraftigt begränsad förmåga att förutspå framtiden blir kollektivet mycket bättre på att förutspå korrekt. Ett annat exempel på detta är när man på mässor kan vinna en tävling genom att gissa antalet godisbitar i en burk. Om man frågar många individer är ett genomsnitt av allas svar ofta närmare det faktiska antalet än individuella kvalificerade gissningar.

Hur gör man för att träna många “medelmåttiga” träd med samma datamängd? Man tillämpar något som kallas bootstrapping. Nya dataset skapas från befintligt dataset. Detta görs genom att kopiera en slumpmässig rad i taget från det ursprungliga datasetet. Man fortsätter med detta tills man har ett lika stort nytt dataset som det ursprungliga datasetet. Observera att när en rad kopieras tas raden *inte* bort från det ursprungliga datasetet. Det innebär att samma rader kan förekomma flera gånger i det nya datasetet. På så sätt skapas svagare

dataset som ligger till grund för hur de individuella trädan i skogen tränas. Då fås träd som inte är bra på prediktioner själva, men som i en större, gemensam kontext kan producera användbara resultat. Detta gör så att varje enskilt träd är unikt och på något litet sätt beskriver en del av sanningen.

De modellparametrar som till störst del avgör hur en tränad modell presterar i praktiken är maxdjupet och antalet träd. Den första innebär att man sätter en gräns på träden i hur djupt de får utvecklas. Detta är viktigt för att modellen ska kunna generalisera bra. Om man tillåter obegränsat djup hade träden matchat träningsdatan exakt, men förmodligen hade modellen inte generaliserats särskilt bra. Det är detta som kallas för att modellen övertränas. Den andra är ganska så tydlig - det är helt enkelt hur många träd som finns i skogen.

5.3 scikit-learn och andra relevanta bibliotek

Implementationskärnan i detta examensarbete är ett Pythonbibliotek vid namn **scikit-learn**. Detta bibliotek innehåller alla steg som krävs för att träna och utvärdera maskininlärningsmodeller. Det mest grundläggande som krävs för att träna en modell finns i Kodblock 1. Relevanta bibliotek importeras. Från ett av dessa biblioteken importeras också ett standarddataset för att illustrera maskininläring. Datasetet delas upp i ett träningsset och ett testset. Modellen skapas (i detta fallet en modell av typen **LogisticRegression**) och tränas med `model.fit()`. Vid denna punkt är modellen nu redo att användas för att förutspå blomsterkategorin av ny blomdata. Rent kodmässigt kan man alltså praktiskt utföra allt som krävs för att träna och utvärdera en modell med mindre än 10 kodrader. I verkligheten krävs ofta väldigt mycket mer än så, exempelvis för att filtrera och transformera om datan.

```
1 from sklearn.datasets import load_iris
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LogisticRegression
4 from sklearn.metrics import accuracy_score
5
6 # Load dataset
7 data = load_iris()
8 X, y = data.data, data.target
9
10 # Split into training and test sets
11 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2, random_state=42)
12
13 # Train model
14 model = LogisticRegression(max_iter=200)
15 model.fit(X_train, y_train)
16
17 # Evaluate model
18 y_pred = model.predict(X_test)
19 accuracy = accuracy_score(y_test, y_pred)
20 print(f'Accuracy: {accuracy}')
```

Kodblock 1: Den kod som minimalt krävs för att ladda in data, träna en modell och utvärdera dess effektivitet i **scikit-learn**.

En annan viktig funktion i **scikit-learn** är klassen **Pipeline**. Med hjälp av denna klass går det att koppla ihop flera datatransformationer i en lång kedja. Det går också att skriva egna transformationslänkar i kedjan. Detta är det primära verktyg som används för att automatisera träning av modeller.

Utöver detta kärnbibliotek finns det ytterligare några Pythonbibliotek som förenklar vardagen avsevärt. **pandas** [1] är det bibliotek som i detta arbete använts allra mest. Det är designat för att man på ett enkelt sätt ska kunna hantera stora mängder data. Grundoperationerna är skrivna i C vilket gör att de i regel är mycket beräkningseffektiva. Alla som ämnar hantera större mängder data och vill ha ett enkelt och flexibelt sätt att interagera med denna datan på skulle ha stor nytta vid att bekanta sig med detta bibliotek. **matplotlib** [9] används flitigt för att illustrera datan. Internt i **pandas** används också **numpy** [11] för att göra matrisberäkningar effektiva. Ibland kommer man i kontakt med råa **numpy**-matriser och vektorer varför det är bra att ha grundläggande förståelse för även detta bibliotek.

En bra introduktion med detaljerad kod till alla dessa bibliotek som är relevanta för maskininlärning finns i ett digitalt tillägg till den bok detta arbete baseras på. [5] För de som önskar tillämpa maskininlärning men som inte har använt dessa bibliotek tidigare är detta en mycket bra källa att börja utifrån.

6 Implementation

I denna del beskrivs varje steg i implementationsprocessen av maskininlärningsmodellen på sågverket. Detta är själva kärnan av arbetet.

Till att börja med ges en översiktlig förklaring över hur vår processkänedom bör inverka på hur modellkonstruktionen sker. Därefter studeras hur processen bäst kan beskrivas genom smart datainsamling. När insamlingsscheman är på plats diskuteras hur en modells struktur skulle kunna utformas för att passa in på den aktuella datan.

För att modellen ska kunna använda sig av datan på ett konsekvent sett tillämpas filtreringar och transformationer på datan för att extrahera så mycket användbar information som möjligt. Sedan konkretiseras vad modellen ska ha som input och output. Därefter väljs vilken typ av modell som ska användas. När en modell väl har valts tillämpas en automatisk slumpmässig utvärdering av modellen för att finna de parametrar som leder till en så bra modell som möjligt.

När det finns modeller som väl förutspår processen byggs ett optimeringsskal utanpå denna så att de optimala maskinparametrarna väljs. Slutligen implementeras detta i sågverkets PLC så att automatisk val av parametrar utförs för varje givet paket.

6.1 Processkänedom

När man som ingenjör ska optimera en process faller det sig naturligt att man behöver ha en grundlig förståelse för processen för att optimeringen ska lyckas. Att tillämpa maskininlärning är inte ett undantag till denna princip. Om man inte förstår processen blir det svårt att veta vilken data som kan vara användbar att samla in, hur denna datan ska filtreras och hur modellen ska utformas för att skapa värde. En detaljerad bakgrund till processen som studeras finns i avsnitt 2.2. I denna del tillämpas den detaljerade beskrivningen och antaganden och avgränsningar görs som kommer att vara användbara i de kommande delarna.

6.1.1 Avgränsningar och antaganden

En plank som färdas genom Elevatorerna ankommer först till Diket vid Elevator 1. Syftet med denna Elevator är att göra en första grövre separation. Sannolikheten för att dubbla brädor uppkommer är förhållandevis hög. Denna första Elevatorn matar sedan Diket vid Elevator 2. Det är plankornas tillstånd efter upphämtning av Elevator 2 som fokuset hamnar på. Efter Elevator 2 finns ingen maskinell möjlighet att separera plankorna.

Utifrån denna beskrivning är det tydligt att Elevator 2 bör vara det största fokuset av arbetet eftersom det är det sista steget plankorna passerar innan de färdas vidare. Vi kommer därför under arbetets gång att hålla oss till att optimera parametrar för Elevator 2.

Ett viktigt antagande är att varje paket som matas in i maskinen kan ses som oberoende av andra paket. Ett paket som körs i maskinen bör rimligtvis

enbart påverkas av maskininställningar, paketdimensioner och tråkighet. Paketet i sig påverkar inte maskinen självt, deras utfall beror bara på tillståndet i nuläget. Detta kommer visa sig hjälpa oss framöver då vi designar maskininlärningsmodellen.

Systemet är i viss mening ganska så trögt och oprecist. Att förändra ställskenorna i maskinen tar tid. I dessa finns också en viss hysteres inbyggd som gör att ställskenorna ofta enbart hamnar i närheten av börvärdet. Utöver detta sker nivåmätningen i Dike 2 med hjälp av en laserdistanzmätare som mäter dikesnivån på en plats. Eftersom bräddor är rektangulärt formade och ligger slumpartat i Diket är detta mätvärde kraftigt varierande. På grund av hur plankorna matas fram via Elevator 1 är det svårt att kontrollera nivån särskilt noggrant. Ibland kommer plankorna i stora klumpar och ibland är det mycket "luft" däremellan. Dessa kombinerade effekter leder till att dikesnivån är en tämligen oprecis parameter. På grund av att maskinen har så pass stor tröghet och låg precision kommer vi att designa en modell som är stabil över tid. Den får inte ge för snabba parameterrekommendationer.

Det finns många dynamiska effekter i processen som inte kan anses normala. En plankan kan kila sig fast och blockera flödet. Det händer också att plankor roteras så att de inte ligger parallellt med varandra vilket även detta stoppar upp flödet. Det kan då se ut som i Figur 19. När det inte finns många plankor kvar i Diket ligger de få som är kvar och "skvalpar". Antalet är så pass litet att trycket är för lågt för att de ska plockas upp av maskinen vid de hastigheter man kör med i standardfallet. Operatören behöver då manuellt justera ner hastigheten för att dessa ska plockas upp korrekt. Det finns säkerligen fler dynamiska effekter som är svårare att förutspå. Därför blir ytterligare en avgränsning att modellen ska agera med steady-state som utgångspunkt. Detta måste beaktas i filtreringen av datan så att så många som möjligt av de datapunkter med dynamiska effekter tas bort.

6.1.2 Börja med slutet

Nästa relevanta frågeställning blir att titta på slutet av arbetet. När modellen är byggd, vad vill vi att den ska kunna åstadkomma i breda termer? Här är det lätt att sväva ut och komma med en lång lista på önskvärda modellfunktioner. Börja istället initiiellt med en eller två funktioner modellen ska ha som hade gett störst optimeringsförbättring. Risken med att bygga en multifunktionell modell är annars att den blir medioker på allting istället för att bli riktigt bra på en eller två saker.

I den aktuella processen är det främst BPM och RDB vi har att jobba med. Målet är att öka BPM samtidigt som RDB minskar, eller åtminstone hålls kvar på samma nivå. Processen är sådan att man i det generella fallet inte kan öka på den ena utan att den andra också påverkas. Frågan är om det finns okända parameterinställningar som hade kunnat bryta denna trend. Förhoppningen är att det är vad en maskininlärningsmodell ska kunna bidra med.

Rent konkret är målet med modellen att den ska kunna förutspå optimala maskinparametrar för aktuellt virke. Eftersom det i praktiken inte är möjligt att



Figur 19: En planka som under processens gång roterats så att den hamnar fel. Detta anses vara en dynamisk effekt och är därför ingenting som modellen tar hänsyn till.

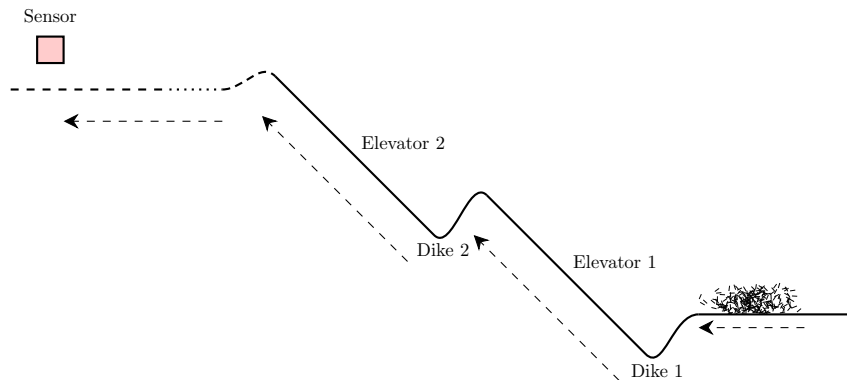
undvika dubbla brädor helt ska operatören kunna ange vilken RDB som man kan tänkas acceptera. Modellen ska sedan välja ut den kombination som håller sig under angiven RDB samtidigt som BPM maximeras.

6.2 Datainsamling

Efter att ha fått en genomgående förståelse för processen och hur vi önskar förbättra den blir nästa fokus att fundera över hur den bäst kan beskrivas med hjälp av data. Även om det redan finns ett befintligt datainsamlingsschema är det bra att begrunda om det inte finns förändringar i datainsamlingen som hade gynnat maskininlärningen. Fler datakolumner som samlas in innebär mer avancerad filtrering senare, speciellt om man vill kunna använda sig av data som samlats in med tidigare insamlingsscheman. Om man till exempel inser att det finns ytterligare en parameter som ska samlas in måste man fundera på hur datan där denna parameter inte finns med ska behandlas. Ska man uppskatta den nya parametern på de gamla dataraderna? Kasta bort datan? Behandla den på ett annat sätt än den nyinsamlade datan? Om det finns en förmodan att ett sådant förändrat insamlingsschema skulle leda till en bättre modell kan det värt mödan.

6.2.1 Datainsamlingens utveckling

I Figur 20 syns processen från sidan. När arbetet påbörjades bestod datainsamlingen av följande: När en planka detekteras vid sensorn loggas alla värden på



Figur 20: De två Elevatorerna sedda från sidan. En hög plankor har precis anlänt till Elevator 1 för separation.

en gång till en databas. Detta betyder att datan som kopplas till varje plank egentligen inte var den data som är relevant för den plankans utfall. När plankan plockades upp vid Diket inpå Elevator 2 hade maskinen vissa momentana värden. Men dessa har förmodligen hunnit ändras till helt andra värden på plankans färd upp mot sensorn. De värden som plockas ut vid loggningen beskriver därför inte det kritiska ögonblicket då plankan plockades upp. För att varje plank ska kopplas till mer rättvisa värden behöver insamlingssättet förändras.

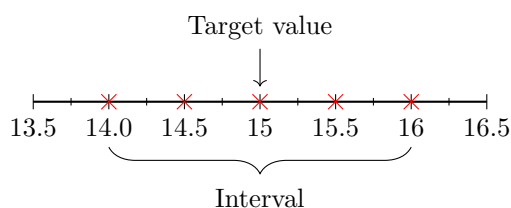
För att ta hänsyn till alla dessa effekter beslutades det att två dataloggningar skulle föras. En logg av diskret typ som tidigare. Den andra loggningen däremot skulle cykliskt samla in maskindata. Genom att separera dessa två loggnings-scheman har vi möjlighet att koppla mer exakt relevant maskindata till varje diskret datarad. I den diskreta loggningen sparar vi sådant som är kopplat till den specifika plankan. Dimensioner, kvalitet, tidpunkt, om en dubbel bräda detekterats, BPM-mål, paketnummer och batchnummer. I den cykliska loggningen sparas allting som har att göra med momentana maskinvärden. Tidpunkt, faktiska elevatorhastigheter, dikesnivåer, ställskenornas nivå samt de maskinvariabler som kontrollerar de faktiska tillstånden. Med hjälp av dessa två insamlingsscheman kan vi skapa ett mer rättvist dataset som beskriver ögonblicket då Elevator 2 plockar upp varje plank. Hur detta görs i praktiken beskrivs i avsnitt 6.4.3.

I detta stadie är det viktigt att samla in datan i ett format med avseende på att tunga beräkningar ska utföras på den framöver. Initiiellt loggades tidpunkter med hjälp av en ISO8601-sträng. [6] Dessa tidssträngar är lätta att visuellt förstå men mycket tidskrävande att utföra beräkningar på, eftersom informationen måste omvandlas från strängar till någon typ av beräkningsobjekt. Lösningen på detta var att samla in tiden i ett format som i Beckhoffs PLC kallas för `T_FILETIME64`. [2] Eftersom detta format är ett enkelt 64-bitars heltal kan beräkningar i tid utföras betydligt snabbare. Det är en fördel att en loggdapunkt i detta format utförs vid insamlingsstadiet eftersom beräkningskraften

vid dataomvandlingen från strängar till heltal snabbt skalas upp i och med att datamängden ökar.

6.2.2 Datavariation

Bra modellprediktioner bygger på bra data. Om maskinen ständigt körs med samma inställningar kommer en eventuell modell inte ha några maskintillstånd att leta i för att finna de optimala inställningarna. För att datakvaliteten ska vara så hög som möjligt måste maskininställningarna ha hög variation. När man ändrar på befintliga inställningar är det naturligt att prestandan sjunker och antalet fel ökar. Detta är en nödvändig del av processen för att datan ska ha den variation som krävs. Det finns sätt att göra detta på som inte inskränker på processen alltför mycket. På sågverket infördes en automatiserad spridning av parametrarna i ett område nära receptvärdet. Om ställskenorna har ett faktiskt börvärde för aktuell dimension på 15 mm, så förändras detta värde inkrementellt mellan 14 mm och 16 mm, med 15 minuter på varje nivå. På så sätt fås en automatisk variation i spridningen utan att den dagliga operationen påverkas i alltför stor utsträckning.



Figur 21: Ett exempel på hur en maskinparameter automatiskt skulle kunna tänkas justeras. I sågverksprocessen används en stegning på 0.5 mm.

Utöver detta infördes också slumpvis variation av parametrar på sågverket, som operatören på plats har möjlighet att pausa om det skulle visa sig att de aktuella värdena leder till alltför dåliga resultat. Detta är ett mer dramatiskt kirurgiskt ingrepp i processen men är samtidigt också det som ger bäst förutsättningar för att en modell ska kunna ge så bra rekommendationer som möjligt.

Det är önskvärt att kunna skilja på de plankor som körts med olika receptinställningar. Därför infördes också true/false-värden som beskriver om maskinen kört med slumpvärden, modellprediktioner eller om den vanliga receptinställningen använts. Detta är viktigt för att möjliggöra framtida resultatkontroller för att ta reda på om modellen faktiskt ökade produktiviteten.

6.2.3 Kontrollera datainsamlingen

När data väl börjar rulla in måste denna kontrolleras så att den stämmer överens med verkligheten. Här är processförståelsen användbar för att identifiera

datavärden som inte är rimliga. Till exempel identifierades att vissa bandhastigheter ibland gick ner till 0 på ett fysikaliskt omöjligt sätt. Efter samrådan med sågverket fann vi att det fanns ett fel i hur hastigheterna skickades vidare och detta korrigerades.

När datan väl hade samlats in under några månader berättade datan för oss att det inte uppkom särskilt många dubbla brädor alls. Detta rimmade inte med operatörernas upplevelser. Efter en dialog med sågverket upptäcktes ett fel i loggningen av dubbla brädor. Konsekvensen blev att mycket av den data som samlats in inte kunde användas i syfte av att ta reda på hur många dubbla brädor som uppkom. Datat kunde däremot fortsatt användas i analyser av BPM.

6.3 Modellstruktur

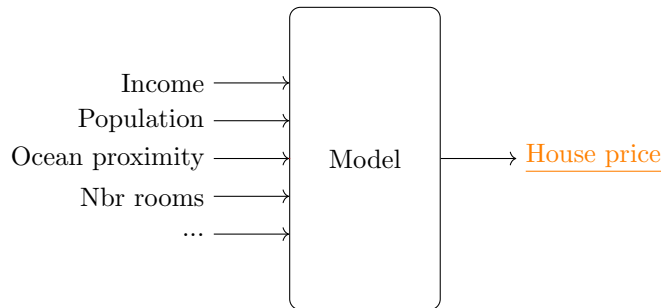
För att det ska gå att välja en lämplig maskininlärningsmodell måste det i detalj vara tydligt vad exakt modellen ska utföra. En modell som bygger på data i form av tydlig input och tydlig output måste anpassas efter den data som finns tillgänglig.

6.3.1 Vanlig modellstruktur

Som nämdes i avsnitt 6.1.2 är målet att modellen ska räkna ut de rekommenderade maskininställningar som leder till det mest effektiva utfallet för aktuellt virke. Inom all typ av modellering måste det finnas data som i någon mening är *korrekt*. Detta krävs för att modellens prediktionsförmåga ska öka. Ett exempel på detta finns i [4] kapitel 2 där man bygger en maskininlärningsmodell för att förutspå fastighetspriser i Kalifornien. För att modellen över huvud taget ska kunna komma med vettiga prediktioner krävs mycket data på faktiska huspriser. Utöver detta måste det också samlas in data som ligger till grund för prediktionerna. Exempelvis används läget, hur nära havet fastigheten ligger, antal rum och så vidare. Genom att träna modellen på denna data kommer vi förhoppningsvis att sedan kunna presentera ny inputdata för modellen och få en prediktion som någorlunda stämmer överens med verkligheten. I detta fall fungerar tillvägagångssättet bra eftersom datan är av tydlig input-output typ. Med andra ord går det väldigt tydligt att koppla den lokala fastighetens omständigheter till dess faktiska slutpris. Frågan blir nu hur detta kan generaliseras till vår situation. I Figur 22 beskrivs denna situation grafiskt.

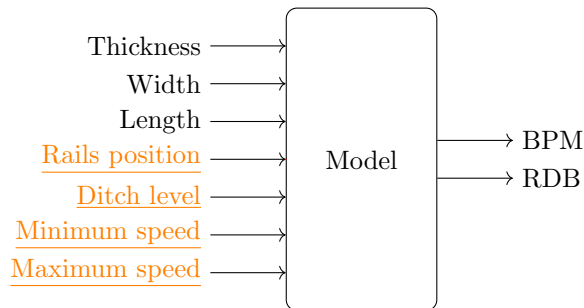
6.3.2 Modellsvårigheter med elevatorsystemet

När modellen utfört en prediktion på givet virke ska den presentera rekommenderade maskinparametrar. Detta är en annan situation jämfört med fallet av huspriser. I elevatorsystemet finns det tillgång till data för dimension och kvalitet innan maskinen körs. Sedan sätts maskinparametrarna. Det är sedan kombinationen av dessa som tillsammans leder till att det blir ett visst utfall - ett specifikt värde på BPM och RDB. Målet är att modellen ska komma med



Figur 22: Grafisk beskrivning av den modell som konstrueras i [4] kapitel 2. Huspriserna är understrukna eftersom målet med modellen är att förutspå huspriser så korrekt som möjligt.

rekommenderade parameterinställningar. Då dessa i sig är en del av systemets input måste detta beaktas i lösningen. Denna situation kan ses i Figur 23. En jämförelse med Figur 22 visar den fundamentala skillnaden i det att för situationen på sågverket är det *parametrar som leder till ett utfall* som ska optimeras. Byggandet av en prediktionsmodell är alltså ett delmål i det övergripande syftet att optimera maskinparametrar.



Figur 23: Situationen för modellen i sågverket. Observera att önskan nu är att optimera modellparametrarna, trots att dessa egentligen är *inputs* till den studerade processen. Modellens möjlighet att förutspå BPM och RDB blir därför ett steg på vägen mot att kunna finna optimala maskininställningar.

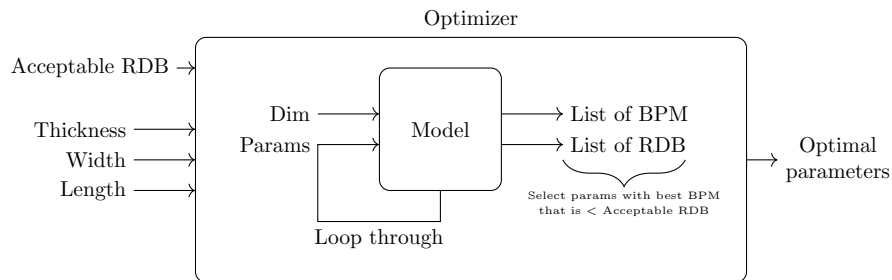
Om det funnits kunskap om korrekta maskininställningar för en viss dimension hade det gått att designa maskininlärningsmodellen på samma sätt som i fastighetsmodellen. Då hade modellen kunnat tränas på virkesinput och som önskad utdata de korrekta maskininställningarna. Problemet är att maskinen *inte nödvändigtvis* är korrekt inställd. De recept som används har tagits fram med hjälp av tumregler och har visat sig fungera bra. Men det finns inget som säger att de är optimala. Om modellen tränas på dessa befintliga recept kommer

resultatet att vara oanvändbart. Modellen hade rekommenderat exakt samma parametrar som redan används i maskinen. Vi måste ta fram ett annat ramverk som är anpassat efter elevatorsystemet.

6.3.3 Modellstrukturen för elevatorsystemet

Även om datan inte innehåller korrekta maskinparametrar är datan ändå korrekt i den bemärkelsen att den visar vilket utfallet blev med given input. Med datan går det alltså att skapa en modell som klarar av att förutspå BPM och RDB givet virkestyp och maskininställningar. När denna modell finns kan man gå “baklänges” genom att utvärdera modellen för olika parameterkombinationer och välja den parameterkombination som resulterar i det mest optimala utfallet. I Figur 24 finns en grafisk beskrivning för hur hela optimeraren fungerar.

Detta är ett beprövat tillvägagångssätt inom industrin. I en artikel optimeras parametrar i en malningsprocess med hjälp av maskininlärning. Datan som samlas in används för att bygga modeller av den faktiska processen. Det är sedan på dessa modeller optimeringen tillämpas. [8] Liknande metodik används också i en artikel där det sker optimering av tillverkningen av rattar till personbilar. [15]



Figur 24: Strukturen för optimeraren som tar hänsyn till att parametrarna som ska optimeras är input till maskininlärningsmodellen.

6.4 Datafiltrering och datatransformering

Nästa fokus är att göra datan mer användbar för modellen genom smart filtrering och transformering. Detta var den del som tidsmässigt var mest krävande. Varje välfungerande modell har bra data som grund.

En viktig grundprincip att följa vid all typ av databehandling är att automatisera allting. Med tiden kommer mängden data att succesivt öka. Då är det fördelaktigt att skapa ett antal regler som datan formateras utifrån så att det blir lätt att transformera om datan. För att förenkla denna process finns `Pipeline`-klassen i `scikit-learn` som beskrivs närmare i avsnitt 5.3.

Det finns ingen perfekt data. All data kan innehålla brus, felaktiga mätvärden, kalibreringsfel, och andra felaktigheter som försvårar möjligheten att göra något användbart med den.

6.4.1 Borttagning av felaktiga datarader

Det är önskvärt att systemet ska vara så robust som möjligt med tanke på att vi matar in rå data. Eventuella felaktigheter ska inte leda till att modellen betar sig på oönskvärda sätt. Datarader med uppenbara felaktigheter i loggningen tas därför bort. Exempelvis tas rader bort som saknar tidsstämpel.

Det finns också datarader med mätvärden som inte är definierade och som därför bör tas bort:

- Elevatorerna kan inte köras med negativ hastighet
- Børhastigheterna till Elevatorerna ska inte vara negativa
- Minhastigheten till Elevator 2 ska inte kunna vara större än maxhastigheten till Elevator 2
- Børvärdet för nivån i Dikena ska inte vara negativa
- Børvärdet för ställskenorna ska hålla sig inom fysikaliskt rimliga gränser

I praktiken ska dessa problem inte kunnat uppstå. Det är däremot viktigt att sträva efter långsiktig robusthet och inte förutsätta att andra system fungerar felfritt. Detta gjordes därför i syfte att göra systemet robust över tid.

6.4.2 Filtrering

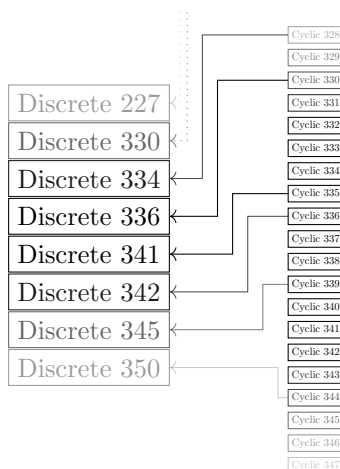
Sedan finns det värden som ska behållas, men i ett filtrerat tillstånd. Till att börja med betraktar vi nivåerna på Dikena. I och med att detta sensorvärde enbart mäts av *en* laserdistansmätare vid varje Dike är detta ett mycket oprecist värde. Sensorn ger också ibland en läsning som fysikaliskt omöjlig kan motsvara verkligheten. Till exempel förekommer det ibland negativa dikesnivåer. För att lösa detta problem implementerades en algoritm som identifierar sådana orimliga mätvärden. Detta görs genom att kolla om mätvärdena antingen är negativa eller om de befinner sig ovanför ett visst ställbart maxvärde.

Mycket tid spenderades på att fundera över hur nivåfiltreringen skulle kunna förbättras. Exempelvis kan ett mätvärde tänkas vara felaktigt även om det hela tiden befinner sig inom de tillåtna gränserna ifall förändringarna sker för fort. Efter ett tag blev slutsatsen att det bästa är att börja med enkla filtreringsscheman, utvärdera resultatet och sedan eventuellt finslipa filtreringen ifall behovet ser ut att finnas. Alla nivåer ovanför en viss gräns sätts till gränsvärdet, och nivåer under 0 sätts till 0. Att börja med enkla filtreringsscheman och sedan utvärdera behovet av vidareutveckling är en viktig allmän princip att följa. Det är annars lätt att fastna i optimeringar som man slutligen inte kommer ha nytta av.

6.4.3 Ihopslagning av datafilerna

Nu har datan fått sin första grundläggande filtrering. Orimliga datarader är borttagna och mätvärdena är mer verklighetstroga. Det är nu dags att transformera datan på smarta sätt inför det att vi tränar vår maskininlärningsmodell.

Till att börja med sker datainsamlingen i två steg i enlighet med beskrivningen i avsnitt 6.2.1. En *diskret* och en *cyklisk* loggfil skapas. Målet är att använda datan i båda databaser på ett smart sätt. Den cykliska datan beskriver *momentana maskintillstånd* medan den diskreta datan innehåller relevant information om *varje enskild planka*. Med andra ord loggas individuell plankdata varje gång en planka passerar sensorn i Figur 20. Den cykliska datan loggas konstant med en cykeltid på 250 ms.



Figur 25: Hur den cykliska datan läggs på den diskreta datan förutsatt en backstep på 6 sekunder. Varje rektangel beskriver en loggrad och heltalen representerar loggradens tidstämpel. Pilarna visar att den relevanta momentana maskindatan från de cykliska dataraderna kopieras över till de diskreta dataraderna. I den verkliga processen används en cykeltid för den momentana datan på 250 millisekunder och inte 1 sekund som grafiken antyder på.

När en maskininlärningsmodell används är det ett krav att det finns välgrupperad inputdata. Dessa två loggfiler behöver sättas ihop till en. Utgångspunkten blir den diskreta datan eftersom det i grund och botten är plankornas resultat vi är intresserade av. På denna diskreta plankdata vill vi sedan knyta de relevanta momentana värdena som gällde vid det kritiska ögonblick då varje individuell planka plockades upp. För att göra detta så enkelt som möjligt stegar programmet ett fast antal sekunder bakåt i tiden och plockar därefter ut de värden som finns där i den cykliska filen. Denna stegning visas grafiskt i Figur 25.

Detta hade kunnat göras mycket mer detaljerat. Alla sträckor och hastigheter mellan sensorn och Diket som matar Elevator 2 är kända. Det är därför teoretiskt fullt möjligt att med hjälp av en plankas tidsstämpel och maskinhastigheterna stega bakåt i tiden för att ta reda på exakt när plankan plockades upp från Diket. En del tid spenderades på att utveckla en sådan algoritm. Det är möjligt att det

hade varit mer fördelaktigt men utifrån att hålla lösningen så enkel som möjligt tillämpades istället filtreringsschemat ovan. Det kommer dessutom att visa sig framöver att lösningen som valdes inte är beroende av exakta momentanvärden.

6.4.4 Den momentana maskindatan

Resultatet är nu att det finns en fil som innehåller all relevant data, både den diskreta plankdatan och den momentana data som beskriver den situation som (ungefärligt) gällde då plankan plockades upp av Elevatorn. Den diskreta virkesdatan är ganska så enkel att implementera i modellen. Det är inte svårt att inse att denna data finns tillgänglig innan maskinen körs och att den har en påverkan på maskinens prestanda. Det svåra blir att ta reda på hur den momentana datan skulle kunna användas på bästa sätt, eftersom denna enbart är tillgänglig först när maskinen körts.

Det finns maskininlärningsmodeller som är dynamiska och kan arbeta på realtidnivå (se kapitel 15 i [4]). Då hade modellen hela tiden kunnat köras på aktuell maskindata. Detta ställer höga krav på både datainsamlingen och på maskinens förändringsbenägenhet. Eftersom processen är diskret i sin natur, och på grund av svårigheten i att reglera exempelvis dikesnivån efter önskemål blev slutsatsen att denna typ av modell förmodligen inte skulle fungera bra.

Fokus flyttades istället till en enklare och "långsammare" modell. Istället för att betrakta datan på planknivå grupperas alla plankor som befinner sig i ett paket ihop. Observera att detta gör datan betydligt mer översiktlig. I någon mening innebär detta att hela händelseförloppet under ett givet paket betraktas som en svart låda. Resultatet som fås ut för hela paketet är det som tas vidare. Eftersom fokus ligger på att ta fram de maskinparametrar som fungerar bäst i steady-state är det rimligt att ett set av parametrar per virkeskombination skulle ge optimal drift. Resultatet av detta blir att optimeringen utförs på genomsnittning BPM och RDB på paketnivå.

Det finns fördelar med att samla in detaljerad data och sedan systematiskt konvertera denna till paket. Den främsta blir att detaljerad data för processen alltid finns tillgänglig. Det går därför att ändra sättet som paketdata konsolideras till en datapunkt. Nackdelen är att datan allt som allt tar upp betydligt mer plats.

6.4.5 Datatransformation till paket

I varje paket finns det hundratals plankor. All denna information ska nu konsolideras till en enda datapunkt. Målet är att behålla så mycket information som möjligt som skulle kunna användas för att förutspå paketets utfall. De värden som lätt följer med är dimension och kvalitet.

Eftersom fokus ligger på att optimera maskinen i steady-state måste så många dynamiska effekter som möjligt sorteras bort. Flera olika strategier utvecklades för detta ändamål. En av dem var att ta bort en viss procentuell andel av plankorna i början och i slutet på varje paket. Detta eftersom det uppstår en större mängd dynamiska effekter då. Däremot gick det inte att se någon

förbättring på modellen med hjälp av denna metod, varför den slutgiltigen inte användes.

Flera av plankornas mätvärden förändras drastiskt under paketets gång. Bandhastigheter och dikesnivåer är exempel på sådana. Dessa är därför svåra att göra någonting användbart med när datan blir så pass översiktlig som den nu är. Parameterinställningarna är däremot i regel konstanta genom ett givet paket. Dikesnivåns börvärde, ställskenornas inställning och bandens max- och minhastigheter håller sig konstanta. Med andra ord blir det lätt att plocka ut dessa värden på paketnivå.

Även om dessa inställningar i regel håller sig konstanta behöver operatören ibland justera dem något, speciellt i slutet av varje paket då mängden plankor minskar avsevärt. För att hantera detta kan vi, varje gång en sådan inställningsändring skett, dela upp paketet där inställningsändringen skedde och betrakta de två delarna som separata paket med olika inställningar. Dessa paket blir också viktade baserat på hur stor del av det totala paketet de mindre paketen utgör. Denna viktning gör att delpaket inte kommer att få orättvist stor betydelse senare då maskininlärningsmodellen tränas. Paketen får inte heller bli alltför små, eftersom det då finns större risk för orimliga resultat. Alla paket som har ett antal plankor mindre än en viss % av paketets totala innehåll kastas därför bort. För större virkesstorlekar där det inte finns så många plankor i varje paket finns också en absolut spärr satt.

Paketet tilldelas samma id som paketets första plankor. Detta id används senare för att skapa tränings- och testset av datan.

BPM för ett paket är det genomsnittliga BPM-värdet för alla plankor i det aktuella paketet. RDB räknas ut genom Ekvation 1.

$$\text{RDB} = \frac{\text{Antal dubbla brädor i paketet}}{\text{Antal brädor i paketet}} \quad (1)$$

Slutligen överförs också två booleanyvärden som beskriver om slumpvärden eller modellen var aktiv under paketets gång. Mer information om detta finns i avsnitt 6.2.2.

6.4.6 Labels

Inom maskininläring innebär "label" det som vi försöker förutspå. I exemplet tidigare där huspriser skulle förutspås ser vi huspriserna som det datasetets "labels".

I detta fall används maskininläring för att förutspå BPM och RDB. Med andra ord är det de faktiska värdena för ett pakets BPM och RDB som är labels. En metod skrevs för att extrahera dessa från inputdatan så att modellen kan tränas med en tydlig input- och output.

6.4.7 Filtrering av BPM

BPM är en av de kritiska värden som modellen ska kunna förutspå. Det är därför nödvändigt att BPM-värden på paketen är verklighetstroga. Att fel

förekommer under elevatorprocessen är vanligt och det leder till en minskning av BPM som inte är att betrakta som steady-state. Om BPM därför skulle sjunka för långt under vanligt rimliga värden kan det paketet betraktas som utsatt för dynamiska effekter. Därför tillämpas en BPM-filtrering på alla paket som avviker med en viss procent från den dimensionens median-BPM. Detta gav kraftigt förbättrade prediktionsresultat.

6.4.8 Filtrering av RDB

Som nämndes i avsnitt 6.2.3 upptäcktes det mitt under projektets gång att det skett ett fel i datainsamlingen med konsekvensen att många datapunkter inte kunde användas i syfte att förutspå RDB. Alla datarader som samlades in innan felet upptäcktes och åtgärdades kastas därför bort.

Eftersom RDB är en relativ parameter måste paket med litet antal brädor behandlas speciellt. Om det förekommer en dubbel bräda i ett paket som enbart har 20 brädor får hela det paketet RDB-värdet 5%, vilket är extremt högt. Paketet anses då vara så pass litet att steady-state aldrig kan uppnås. Om man inte skulle ta hänsyn till detta hade datan alltså innehållit stora outliers som skulle försämrat modellen.

6.4.9 Tränings- och testset

En metod skapades för att dela upp datan i ett träningsset och ett testset. Information för den intresserade läsaren varför detta görs finns i avsnitt 5.1.2. Uppdelningen görs deterministisk genom att paketens id används.

6.5 Val av input och output

I avsnitt 6.3 nämndes att maskininlärningsmodellens ultimata syfte är förutspå optimala maskinparametrar. Ett delmål för att uppnå detta ultimata mål blir att förutspå BPM och RDB. I detta syfte skapas två olika modeller. En modell vars jobb är att förutspå BPM, och en annan vars jobb är att förutspå RDB. Båda modeller har samma input men tränas för att förutspå olika input.

I varje paket har virket en viss kvalitet. Detta skulle definitivt kunna tänkas ha en påverkan på hur ofta dubbla brädor uppkommer och hur effektivt maskinen kan arbeta. I ett initieellt skede togs däremot beslutet att bortse från detta. Dels för att det finns bristande mängd data vilket hade gjort att det inte funnits så mycket data i varje kvalitetskategori, men också för att dimensionen troligtvis är de lågt hängande frukterna som kan producera störst effekt med minst ansträngning.

Maskinparametrarna ska också skickas in till modellen då det slutgiltigen är dessa som arbetet ska optimera. Här finns en del funderingar att göra. Det finns fyra parametrar som skickas in: maxhastigheten, minhastigheten, börnivån på Diket och börnivån för ställskenorna. Detta är parametrar som kan sättas godtyckligt. I och med att dessa parametrar är börvärden är frågan om det går att använda sig av de faktiska momentana värdena på något smart sätt.

Hastigheterna för varje planka varierar storligen under paketets gång. Samma situation gäller för dikesnivån. Därför är det inte lämpligt att fundera över den momentana hastighets- och dikesnivådatan. Däremot är det intressant att titta på ställskenornas faktiska lägen. Eftersom dessa ofta skiljer sig åt från sina börvärden på grund av den inbyggda hysteresen (se avsnitt 2.2) blir datan mer rättvis om ett genomsnitt av ställskenornas faktiska värden används istället för börvärdet. Detta är lämpligt att göra för ett visst paket, eftersom ställskenornas inställning är statisk för varje paket.

Sammanfattningsvis skickas följande parametrar in i maskininlärningsmodellen:

- Tjocklek, bredd och längd
- Maxhastighet och minhastighet
- Börvärdet för dikesnivån
- Faktiskt genomsnittsvärde för ställskenorna. Detta räknas ut genom att ta genomsnittet av ställskenans värde för varje plank i paketet.

Två modeller konstrueras. En som förutspår BPM och en som förutspår RDB. Dessa modeller ligger sedan till grund för optimeringen av maskinparametrarna.

6.6 Modellval

Då maskininläring är en ganska oprecis vetenskap som baseras på experiment snarare än analytiska slutsatser är detta också metodiken som tillämpas vid val av maskininläringmodeller. Man får testa sig fram och se vilka modeller som ser ut att prestera bra i praktiken. Detta är också precis vad som utförts. Ett antal modeller i standardformat (utan att skruva på några modellinställningar) testades på datan i syfte att förutspå BPM för att få en allmän bild av vilken modell som kan tänkas prestera bäst. Modellens prestanda jämförs sedan med de faktiska värdena. Resultatet av detta finns i Figur 26.

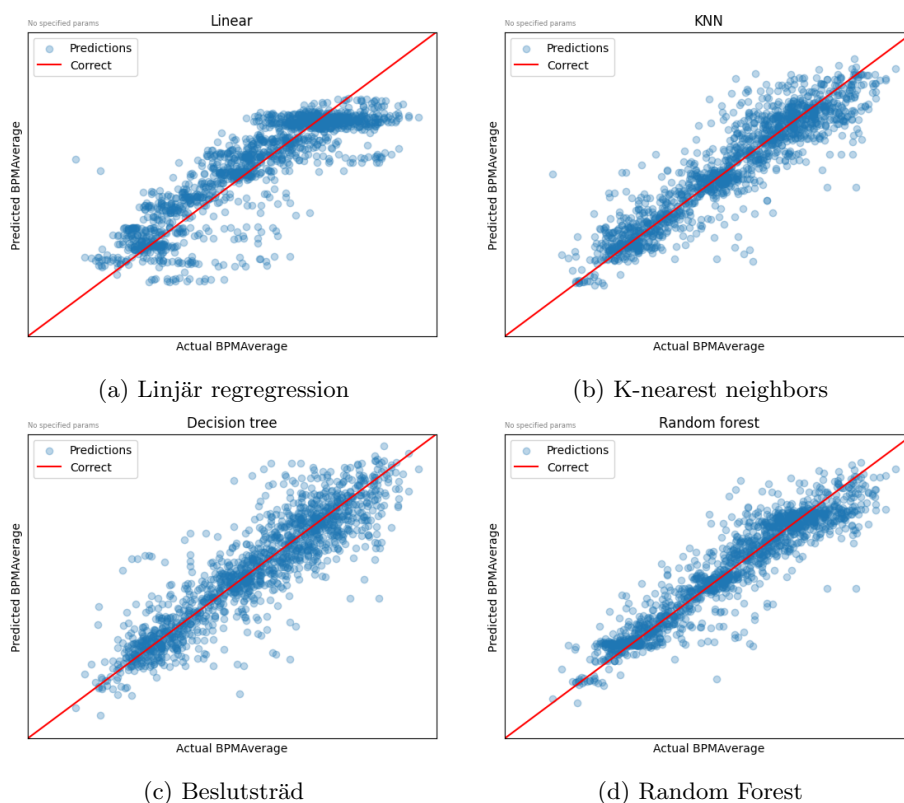
Som förväntat presterar den vanliga linjära regressionsmodellen sämst av alla. Hela syftet bakom att använda sig av maskininläring i detta fall är för att processen är olinjär. Således fås tydligt definierade horisontella linjer.

De två följande modellerna KNN och beslutsträd har större spridning i sina datapunkter och har därför inte samma problem med horisontella linjer som den linjära modellen har. Det finns däremot många punkter som ligger långt från linjen. Maskininlärningsmodellen KNN har inte beskrivits i detta arbete. Hur modellen fungerar i detalj är inte viktigt att förstå här. Det enda relevanta är att KNN är en annan maskininlärningsmodell som de andra modellerna kan jämföras med.

Den modell som fungerade allra bäst är Random Forest. De horisontella linjerna undviks fortsatt samtidigt som dataklustern ligger nära linjen. Eftersom denna modell ser ut att prestera bäst i detta initiala test är det också den modell som väljs ut för vidare testning.

Det bör poängteras att testsetet i regel inte bör användas för att utvärdera hur bra en modell presterar. Detta är för att en modell presterar olika bra baserat på vilken del som väljs ut som testdata. I detta fallet ska vi enbart översiktligt säga något om hur de olika modellerna kan tänkas prestera. För slutgiltig bedömning av hur bra den valda modellen är kommer korsvalidering att utföras. För läsaren som inte stött på korsvalidering (cross-validation) innan hänvisas till avsnitt 5.2.1.

Eftersom mycket RDB-data behövde kasseras på grund av ett insamlingsfel valdes Random Forest för att träna en modell på denna data också, utan vidare analys av olika modelltyper. I framtiden när mer data samlats in kan det vara nyttigt testa olika modeller på denna data också.



Figur 26: Prestandan av olika baselinemodeller. Inga modellinställningar skruvades på. Varje datapunkt är ett paket. x-axeln beskriver faktisk BPM, y-axeln beskriver modellens förutspådda BPM. Således hade alla datapunkter legat på linjen om modellerna varit perfekta.

6.7 Automatisk utvärdering av modell

Efter att ha beslutat att Random Forest är den modell som ska studeras vidare ligger fokus nu på att denna ska optimeras. Hur den fungerar i detalj beskrivs i avsnitt 5.2. Kortfattat finns det främst två parametrar som kan justeras som förändrar modellens prestanda. Antalet beslutsträd och hur djupa de träden tillåts att vara. Genom att skruva på dessa värden går det att sätta modellens komplexitet. Om modellen är för komplex kommer den att spegla träningsdatan väldigt väl men inte vara bra på att generalisera. Om modellen är för enkel kommer den inte att hitta relevanta samband för processen.

Som nämnts tidigare är maskininlärning en experimentell vetenskap i den bemärkelse att man kastar saker på väggen och ser vad som fastnar. Samma metodik tillämpas vid val av modellinställningar. Eftersom processen ska vara så automatiserad som möjligt behövs ett sätt att utvärdera hur bra en viss modell presterar. I praktiken utförs detta genom att mäta modellens prediktionsfel. I detta arbete har vi valt att använda oss av det absoluta prediktionsfelet eftersom det är enklast att förstå vad det innebär i praktiken. Det är däremot värt att testa ett antal olika sätt att mäta felet på då olika metoder kan vara olika bra lämpade för olika dataset.

För att finna optimala parametrar användes sedan klassen `RandomizedSearchCV` i `scikit-learn`-biblioteket. Fördelen med denna genomsökningsalgoritm är att man har kontroll över hur många olika modeller som tränas. Man anger ett intervall av parametrar som ska sökas igenom och så väljer algoritmen parametrar i intervallet. Både modeller av låg komplexitet och hög komplexitet kördes.

En nödvändig sak att beakta är modellstorlek. Om man tänker sig att ett träd tillåts att ha ett djup på 30 innebär detta att varje träd, om det fylls helt och hållet, kommer att bestå av $2^{31} - 1 = 2\,147\,483\,647$ noder. Under förutsättning att en nod kräver 1 byte med plats (i verkligheten är det förmodligen mycket större) innebär detta att varje träd maximalt kan ta upp 2.15 GB. Om man också tar hänsyn till att varje skog består av flera träd blir det snabbt tydligt att modellstorleken kraftigt kan skjuta i höjden.

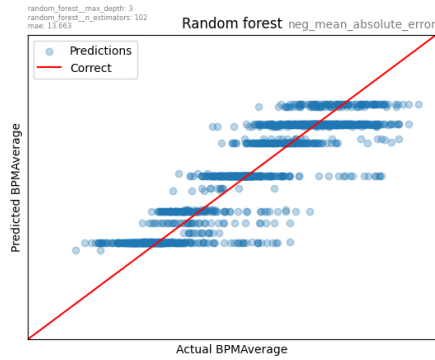
Nu kommer träden med allra största sannolikhet inte att vara riktigt så fullt ifyllda. Det är däremot viktigt att ta hänsyn till att val av en beräkningsmässigt dyrare modell måste kunna motiveras i form av tydlig prestandavinst.

6.7.1 BPM

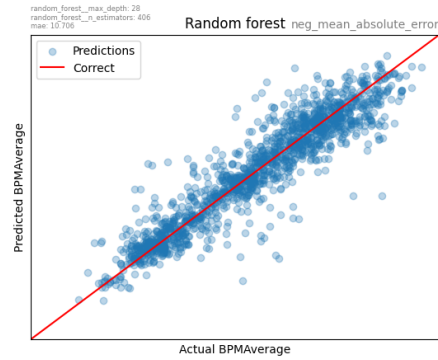
I Figur 27 och Figur 28 finns bilder som visar resultatet av ett antal körningar med `RandomizedSearchCV`. I Tabell 3 presenteras numeriska resultat från utvärderingen.

I Figur 27a används låga modellvärden. Det betyder att modellens komplexitet också är låg. Detta avspeglas i det faktum att det finns många horisontella datalinjer. En lite mer avancerad modell med större djup finns i Figur 27b. Den har mer av den typ av utseende som är önskvärt, med större spridning samtidigt som punkterna håller sig koncentrerade nära linjen. Den modell som tas vidare för att användas i den faktiska processen finns i Figur 27c.

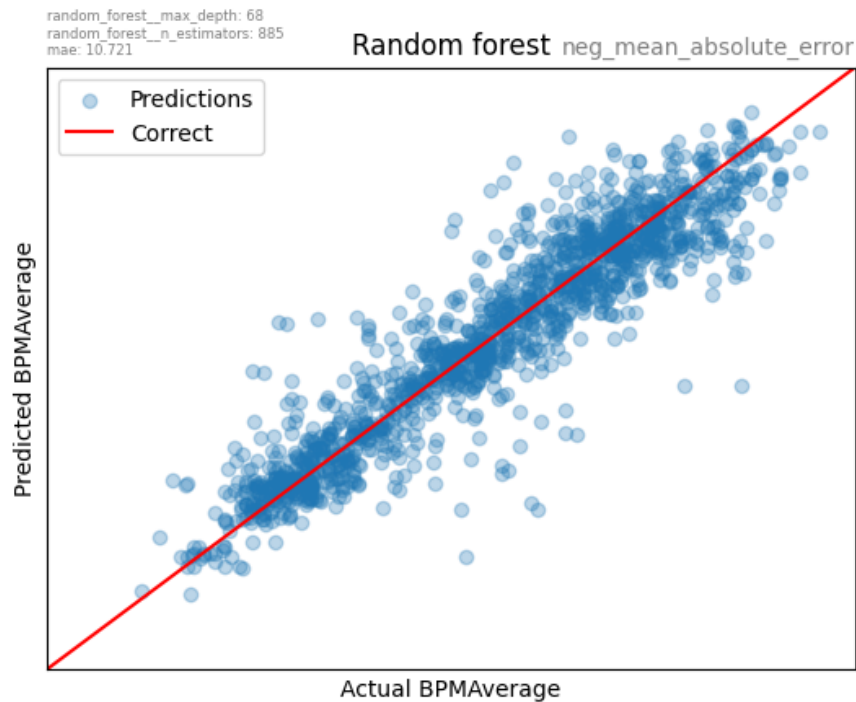
Det kan också vara intressant att se resultatet av att låta djupet ha ett



(a) Låga modellvärden. Djup: 2-20, antal träd: 50-200, cv: 10. Bästa: djup 3 med 102 träd.

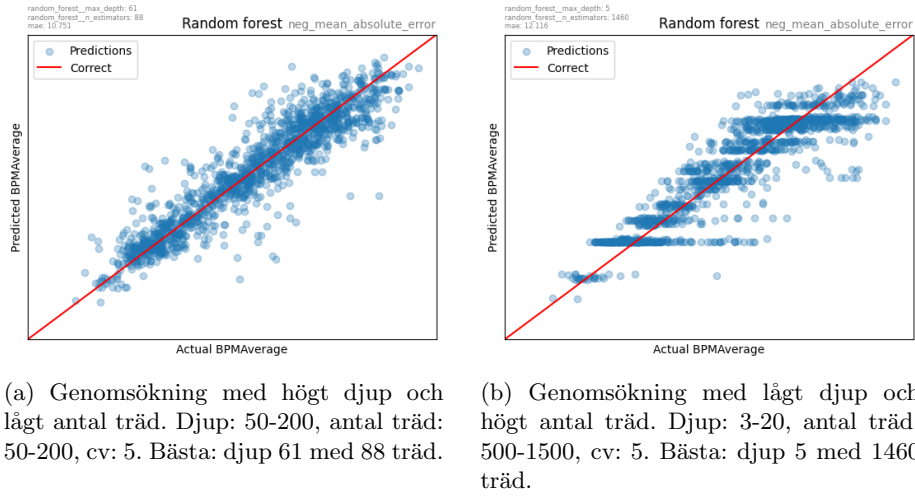


(b) Modellvärden i mitten. Djup: 20-50, antal träd: 200-500, cv: 5. Bästa: djup 28 med 406 träd.



(c) Höga modellvärden. Djup: 30-100, antal träd: 500-1500, cv: 3. Bästa: djup 68 med 885 träd.

Figur 27: Resultatet av att ha kört RandomizedSearchCV på vår Random Forest-modell med olika intervall av värden på träddjup och antal träd i skogen. I alla fall har 100 olika parameterkombinationer testats.



Figur 28: Alternativa intervall för RandomizedSearchCV.

max depth	forest size	fit time	inference time	mean score	std
3	102	0.67	0.02	15.63	2.42
28	406	7.69	0.11	17.60	1.63
68	885	11.18	0.15	16.68	1.38
5	1460	11.61	0.26	15.76	1.02
61	88	2.03	0.03	17.51	1.72

Tabell 3: Modellprestanda för genomsökningarna som finns i Figur 27 och Figur 28. “mean score” anger genomsnittlig avvikelse från korrekt BPM. Denna inkluderar korsvalideringen. Således är lägre värden bättre. “std” är standardavvikelsen från det genomsnittliga inferensresultatet då olika korsvalideringar kördes. “fit time” och “inference time” anger hur lång tid den aktuella modellen tar att träna respektive utvärdera, angivet i sekunder.

högt värde och mängden träd vara litet, och tvärt om. Detta ses i Figur 28. I Figur 28a hålls djupet högt och det används ett litet antal träd. I Figur 28b är situationen den motsatta. Det syns tydligt att ett lågt djup i regel är den främsta boven till de horisontella linjerna.

För att avgöra vilken modell som lämpar sig bäst analyseras en kombination av valideringsresultat och de grafer som beskrivits. Modellen ska vara så bra på att förutspå som möjligt men också ha en bra spridning i de prediktioner som den kommer med. I Tabell 3 finner vi att de modeller med bäst genomsnittligt resultat är de vars djup är lågt. Genom att granska Figur 27a och Figur 28b ser vi emellertid att de har problemet med de horisontella linjerna. Även om de är bäst ur genomsnittlig prediktionssynpunkt är modellerna inte lämpade för att komma med bra optimeringsprediktioner. Resterande modeller har bra spridning på datan. Den parameterkombination som presterar allra bäst är också den med störst djup och störst antal träd. I medel lyckas den marginellt bättre än den betydligt mindre modellen. Vinsterna av att välja en beräkningsmässigt dyrare modell är förhållandevis små och kan inte anses berättiga valet av en sådan. Ytterligare en genomsökning utförs därför mellan låg- och medelvärden på modellparametrarna för att täcka det genomsökningsfönstret mer exakt.

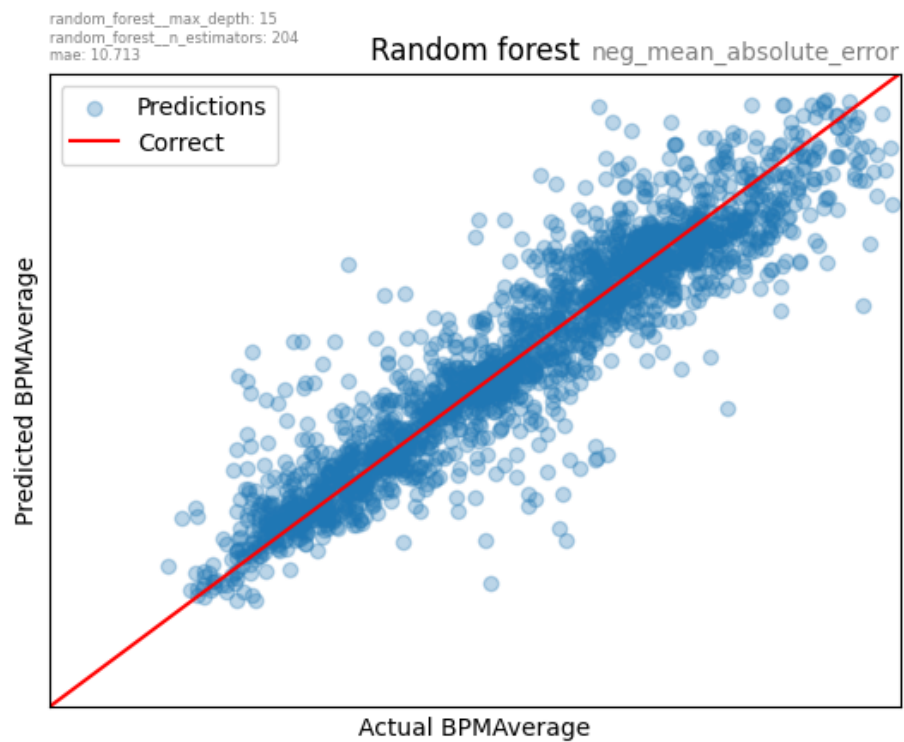
I detta syfte körs ytterligare en genomsökning med ett djupintervall mellan 15-40 och ett trädintervall mellan 200-400. Resultatet av detta finns i Figur 29. De 10 bästa resultaten av denna genomsökning presenteras i Tabell 4.

Resultatet visar på att en mindre komplex modell presterar bra nog för behoven i denna tillämpning. Det faktum att de kombinationer som presterade allra bäst också ligger i nedre delen av parameterintervallen tyder på att ytterligare vinster skulle kunna fås genom att sänka den undre gränsen. Men kom ihåg att de tidigare genomsökningarna visat att detta leder till horisontella linjer i modellens prediktioner vilket inte är önskvärt. I Figur 29 finns inga tecken på de oönskade horisontella linjerna varför de slutgiltiga modellparametrarna blir ett djup på 15 med 204 träd.

6.7.2 RDB

På grund av plankors diskreta natur är RDB-värdet något svårare att beakta vid optimerandet av en maskininlärningmodellen. BPM är ett kontinuerligt värde från 0 och uppåt. RDB-värdet har en naturlig spridning från paket till paket. Två identiska paket med samma maskininställningar kan få helt olika RDB-värden. Det är att förväntas. Även en väl inställd modell kommer därför ha viss naturlig spridning.

I Figur 30 går det att se den bästa modellen som hittades vid genomsökningen av olika modellparametrar. Här blir det tydligt att RDB fundamentalt skiljer sig från BPM i det att RDB-värdena har en naturlig variation. I Tabell 5 presenteras de tio bästa resultaten från genomsökningen. Utifrån tabellen går det avläsa att det enbart är ytterst små vinster som kan fås genom att skruva på modellparametrarna. Detta tyder på att ett fundamentalt annorlunda tillvägagångssätt skulle behövas vid transformationen av datan. På grund av att enbart små vinster fås vid val av komplexare modell väljs den enklare modellen med ett djup



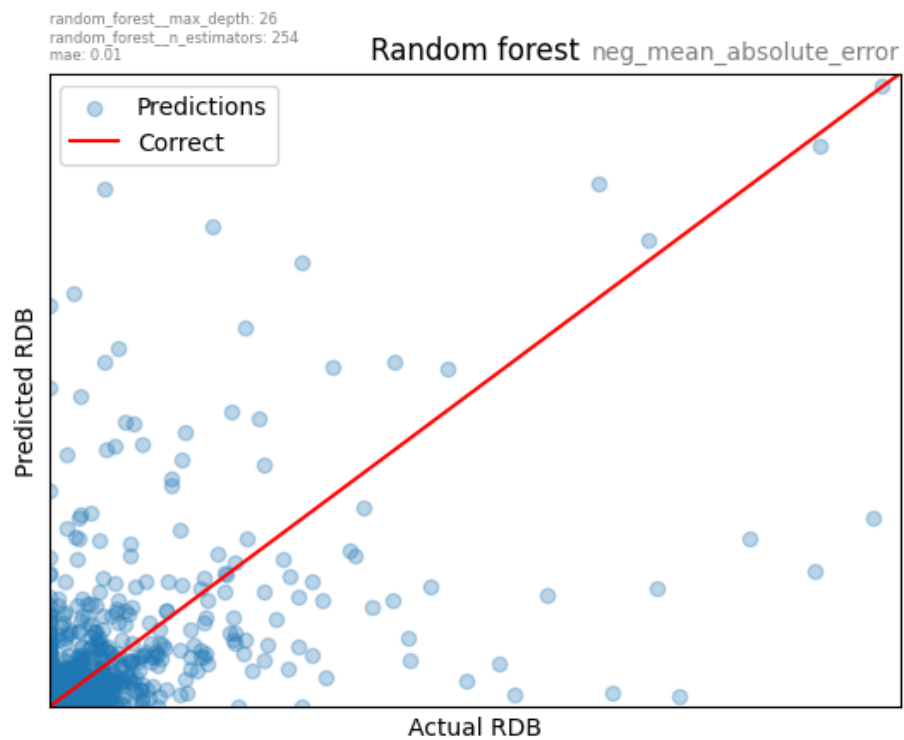
Figur 29: Slutgiltig korsvalidering av träningsdatan för att finna optimala BPM-modellparametrar. Totalt 100 genomsökningar utfördes. Djup: 15-40, antal träd: 200-400, cv: 5. Bästa: djup 15 med 204 träd.

max depth	forest size	fit time	inference time	mean score	std
15	204	4.02	0.056	17.48	2.99
15	238	8.56	0.111	17.55	3.05
15	288	8.18	0.099	17.60	2.99
37	263	9.14	0.122	17.61	2.99
15	300	6.12	0.075	17.61	3.08
15	247	8.73	0.094	17.62	2.99
16	369	6.99	0.091	17.62	3.03
35	302	6.52	0.082	17.63	2.97
22	354	7.68	0.092	17.63	3.04
16	333	11.31	0.165	16.64	3.06

Tabell 4: BPM-modellprestanda för de tio bästa resultaten av genomsökningen som finns i Figur 29. “mean score” anger genomsnittlig avvikelse från korrekt BPM. Denna inkluderar korsvalideringen. Således är lägre värden bättre. “std” är standardavvikelsen från det genomsnittliga inferensresultatet då olika korsvalideringar kördes.

på 18 och en skogstorlek på 249 som den bästa.

På grund av tidsbrist identifieras detta som en framtida potentiell förbättring.



Figur 30: Slutgiltig korsvalidering av träningsdatan för att finna optimala RDB-modellparametrar. Totalt 100 genomsökningar utfördes. Djup: 15-35, antal träd: 200-400, cv: 5. Bästa: djup 26 med 254 träd.

max depth	forest size	fit time	inference time	mean score	std
26	254	5.93	0.093	0.03236	0.01547
18	249	3.48	0.053	0.03240	0.01576
31	251	3.90	0.063	0.03247	0.01534
27	261	5.43	0.081	0.03250	0.01506
22	362	5.01	0.074	0.03252	0.01575
34	295	4.56	0.065	0.03253	0.01573
19	330	6.60	0.094	0.03253	0.01572
19	298	6.05	0.093	0.03256	0.01569
18	402	7.01	0.105	0.03258	0.01528
18	212	2.70	0.041	0.03261	0.01584

Tabell 5: RDB-modellprestanda för de tio bästa resultaten av genom-sökningen som finns i Figur 29. “mean score” anger genomsnittlig avvikelse från korrekt RDB. Detta är ett genomsnittligt resultat från alla korsvalideringar. Lägre värden är bättre. “std” är standardavvikelsen från det genomsnittliga inferensresultatet då olika korsvalideringar kördes. De extremt små skillnader som fås när modellinställningarna skruvas på tyder att datan i sig är begränsad och att det inte går att uppnå ett bättre resultat utan vidare filtrering.

6.8 Optimering av maskinparametrarna

Efter att ett modellval har utförts och dess parametrar nu optimerats är det dags att bygga det övergripande ramverket som rekommenderar maskinparametrar.

Två modeller har byggts. En modell som förutspår BPM och en som förutspår RDB givet virkesdimensioner och maskininställningar. Det slutgiltiga målet är att optimera maskininställningarna. Som Figur 23 antyder leder dessa inställningar till ett visst utfall för processen, vilket betyder att de också är inputs till modellen. För att dessa ska kunna optimeras byggs därför ett skal utanpå modellen som testar olika parametrar för att finna vilken kombination av parametrar som resulterar i bäst kombination av BPM och RDB. En schematisk beskrivning av detta finns i Figur 24.

6.8.1 Optimeringsalgoritmen

Eftersom att det enbart är fyra maskinparametrar som ska optimeras togs beslutet att det skulle vara tillräckligt med någon typ av brute-force genomsökningsalgoritm. Detta hade snabbt blivit omöjligt om antalet maskinparametrar som ska optimeras är stort. Men i detta fall är det en bra utgångspunkt.

Optimeringen sker i två steg. Först görs en grov genomsökning med stora intervall mellan parametervärdena, så att hela vidden av intervallen för parametrarna blir genomsökt. Sedan väljs de kombinationer ut med störst förutspådd BPM samtidigt som RDB hålls under det acceptabla värdet. Dessa går sedan vidare till nästa optimeringsstadie, där en finare genomsökning görs. När denna mer detaljerade sökning utförts filtreras återigen RDB-värden bort som inte uppfyller angivet krav, varefter den maskininställning med bäst BPM väljs ut.

En viktig aspekt av denna genomsökning är att i den mån det är möjligt begränsa genomsökningsintervallen. Här är kunskap om processen användbar. Till exempel används brädornas bredd för att finna en bra nivå på ställskenan att utgå ifrån i genomsökningen. Om en plankas bredd ligger på 16 mm finns det ingen anledning att kolla om ställskenans inställning över 20 mm är mer effektiv eftersom det går att fysikaliskt inse att detta leder till kraftigt ökad RDB. I den finare optimeringen ändras inte ställskenornas värden för att spara beräkningskraft.

6.8.2 Parallellisering

Eftersom varje iteration innebär att maskininlärningsmodellen utvärderas blir detta tidskrävande att utföra när kombinationen av parametrar som ska utvärderas är stort. Detta är dock en beräkning som lämpar sig utmärkt för flerkärnade processorer. När en statisk modell väl har tränats ändrar den sig inte när den utför prediktioner. Att anropa samma modell från flera pågående processer samtidigt är därför helt och hållet trådsäkert. Detta implementerades därför med hjälp av pythonbiblioteket `concurrent`, där processer och trådar kan startas manuellt.

Det finns en hel del overheadförluster med parallella beräkning som måste beaktas. I detta fall tar varje modellprediktion ungefär 0.1 sekunder att ut-

värderas. Om inte overheadkostnaden beaktas kommer det inte vara effektivt med parallella beräkningar. Problemet är att det finns *många* beräkningar som behöver utföras där varje beräkning tar omkring 100 ms. För att minska overheadkostnaden delas alla beräkningar upp i så många processorkärnor som avses användas och så får varje process en klump beräkningar att utföra. Annars hade processerna plockat en beräkning i taget vilket hade lett till större overheadförluster.

Innan detta utfördes var inte minihastigheten en maskinparameter som optimerades på grund av tidsbrist. I och med införandet av parallellisering blev optimeringen så pass effektiv att det fanns tid att även optimera minihastigheten.

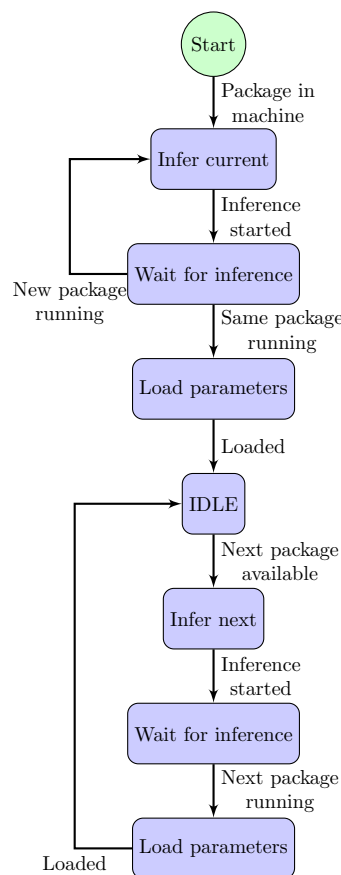
6.9 Implementation i PLCn

En övergripande schematisk bild över hur maskininlärningsmodellen implementerats i sågverkets PLC finns i Figur 31. Här kommer funktionaliteten kortfattat att beskrivas.

När programmet startar är det första den gör att köra en modellinferens på nuvarande paket i maskinen. Detta gör att PLC-programmet kan startas när som helst, även mitt i en körning av ett paket. Som nämnts tidigare tar inferensen viss tid att utföra. När inferensen är utförd kontrolleras att samma paket som inferensen gjorts på fortfarande körs i maskinen. Om det gör det laddas de rekommenderade inställningarna. Annars körs en inferens på det nya paketet.

När inferens är klar för det nuvarande paketet börjar modellen att arbeta i förväg. Nästkommande pakets värden tas in och en inferens körs på detta medan maskinen tuggar igenom nuvarande paket. Detta gör så att så fort nästa paket har laddats in i maskinen laddar programmen in de färdigberäknade rekommenderade maskinparametrarna.

Själva inferensen görs i ett separat pythonprogram som startas av PLCn genom att anropa kommandotolken.



Figur 31: Tillståndsdigram för PLCn

När inferensen är färdig skrivs resultatet till en json-fil som sedan avläses av PLC-programmet och skickas som aktuella rekommendationer. Utöver de aktuella rekommendationerna skickas även modellens förutspådda BPM och RDB över i syfte att ge operatören en bild av vad som kan förväntas med aktuella inställningar.

Modellen skulle kunna utvärderas helt offline genom att generera aktuella parametrar för olika dimensioner som sedan fungerar som recept för processen. I den aktuella tillämpningen körs en optimeringsvända av modellen inför varje paket. Fördelen med detta är att många timmars skapande av tabellerna besparas vilket möjliggör fortare utbyten av modellen i takt med att dessa tränas om.

6.10 Förbättrad utvärdering av resultatet

Som nämnts fokuserar detta examensarbete på att leverera bra prestanda i kontinuerlig, steady-state drift av maskinen. I utvärderandet av hur bra modellen fungerar i drift räcker det inte att titta på maskinens prestanda i det kompletta systemet eftersom omgivande maskiner också har en effekt på dess prestanda. Om maskinen plockar bitar mer effektivt ökar sannoliken för att maskinerna efter istället blir till flaskhalsar vilket sänker BPM för den studerade maskinen, trots att den egentligen plockade bättre. På grund av detta måste ett så rättvist och isolerat resultat som möjligt tas fram.

I avsnitt 2.2.6 förklarades hur maskinen i sitt nuvarande skick tar hänsyn till maskinstopp. Varje gång det går mer än 5.5 sekunder mellan två plankor sker en paus i uträknandet av BPM. Hade detta inte varit fallet så hade BPM sjunkit och varit 0 en minut senare. Det finns däremot fall där detta tidsintervall är en oproportionellt hård bestraffning. Om det finns ett hålrum, följt av enbart en plank, följt av hålrum igen kommer detta dra ner BPM-värdet avsevärt, trots att själva maskinstoppet skulle kunna bero på externa omständigheter.

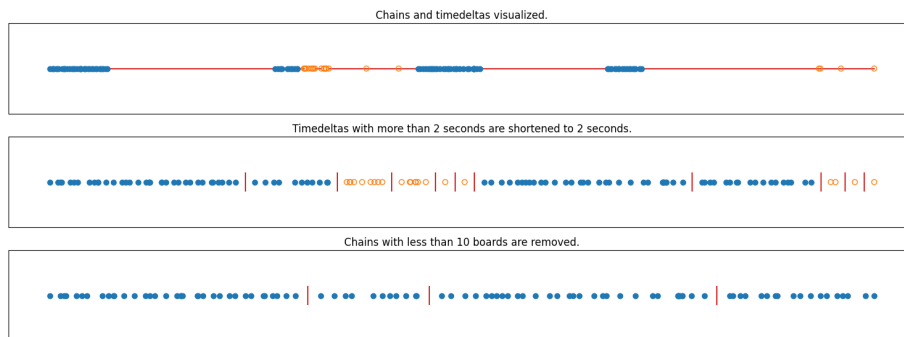
För att få ut en mer rättvis BPM räknas denna om för varje paket. I omräkningen används följande filtreringsprinciper:

- BPM-beräkningen av ett paket börjar när första plankan för det paketet loggats, och slutar när sista plankan i paketet loggats.
- Varje paket delas upp i ett antal "kedjor". Varje kedja innehåller plankor som plockats upp i steady-state drift.
- En ny kedja bildas när en plank inte loggats på 2 sekunder eller mer.
- Mellanrum mellan kedjor kortas ner till 2 sekunder.
- Kedjor med färre än 10 plankor tas bort. Kedjor som inte kommer upp i detta antal kan inte antas köras i steady-state.

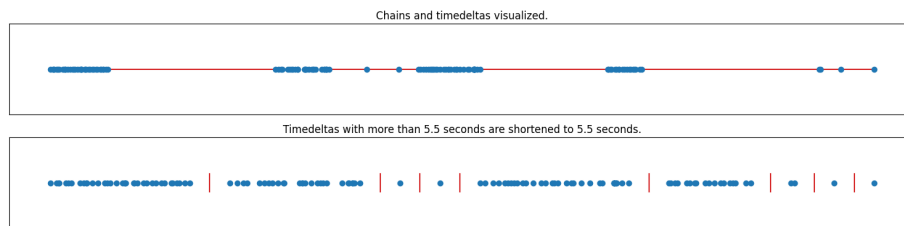
Anledningen till att mellanrum mellan kedjor kortas ned är för att maskinstopp i regel inte är Elevatorernas fel utan kan istället härledas till dynamiska effekter. Genom att korta ner dessa tidsmässiga pauser kan man få ett mer rättvist steady-state perspektiv.

Genom att både filtrera paket där modellen har använts och där den inte har använts med samma filtreringsschema går det är att avläsa om maskinen plockat bättre eller sämre oberoende av externa faktorer.

I Figur 32 går det att se hur denna filtrering utförs i praktiken. Grafen längst upp är alla plankor som passerat sensorn för ett visst paket. x-axeln anger tid. I den första raden går det att se maskinstopp som är ganska så långa. Dessa ska maskinen inte bestraffas för, så de kortas ner till 2 sekunder i enlighet med vad som sagts tidigare. Resultatet av detta ses i den andra grafen. Slutligen tas plankkedjor med färre än 10 plankor bort. Det slutgiltigt filtrerade resultatet syns i grafen längst ner. Samma paket med det insamlingsschema som finns i sågverket idag illustreras i Figur 33. Observera att det finns många fler hålrum i jämförelse med de filtrerade paketen.

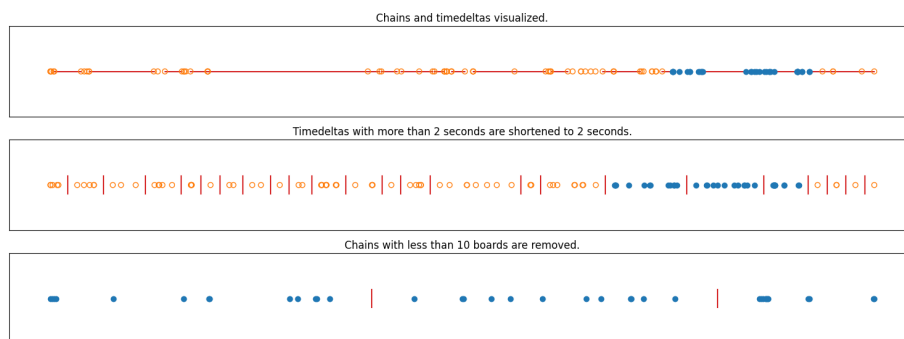


Figur 32: Filtrering av paket i syfte att sätta ett mer rättvist och oberoende BPM-värde på paketet. Alla x-axlar anger tid. Grafen längst upp ger en bild av hur alla plankor i ett paket passerat sensorn ofiltrerat. Den mellersta grafen visar hur paketet förändras då mellanrum längre än 2 sekunder kortas ner till 2 sekunder. Den sista grafen visar hur paketet förändras då plankkedjor med färre än 10 plankor tas bort. Observera att alla tre grafers x-axlar har olika tidsskalor då hålrum och kedjor med få plankor tas bort.



Figur 33: Samma princip och paket som i Figur 32 men med de grundinställningar som finns på sågverket idag. Observera att antalet hålrum i grafen längst ner här är betydligt fler och större än i Figur 32.

Filtreringsschemat är emellertid inte perfekt och det finns rum för förbättringar. I Figur 34 finns ett exempel på ett paket där nästan hela paketet filtreras bort. Paketmellanrummet på 2 sekunder lämpar sig inte här.



Figur 34: Samma princip som i Figur 32 men för ett paket där filtreringen inte fungerar bra.

7 Resultat

7.1 Modell och modellstruktur

Maskininlärning är en högst experimentell vetenskap. När man funderar över vilken modell som ska användas bör man därför utföra valet experimentellt. Detta görs genom att träna flera olika standardmodeller och med hjälp av korsvalidering ta reda på vilken som i genomsnitt presterar bäst. Vilken modell som lämpar sig bäst kan variera från tillämpning till tillämpning men visade sig i detta fall vara modelltypen Random Forest.

I industriella optimeringsprocesser där man inte vet vad det innebär att ställa in en maskin optimalt innebär detta också att datan innehåller bristfälliga maskininställningar. Därför går det inte att träna en modell direkt på dessa maskininställningar. Modellstrukturen behöver göras något mer avancerad än så. Detta löses genom att dela upp modellen i två steg.

I sågverksprocessen är det första steget att skapa en modell av processen som, med given processdata och maskinparametrar ger en prediktion på vad utfallet kommer att vara. För varje kombination ska modellen förutspå vilken BPM och RDB som de nuvarande inställningar rimligtvis bör leda till. Det andra steget blir sedan att söka igenom denna modell av den verkliga processen för att finna de maskininställningar som bör leda till mest optimalt utfall.

7.2 Modellprestanda

Med ett fokus enbart på det befintliga resultatmätningsschemat (BPM) som används i sågverket kunde inga förbättringar märkas med hjälp av modellen. Vid denna punkt blev fokuset att isolera maskinens prestanda och se vilken plockprestanda den har när den väl körs kontinuerligt. Genom det filtrerings-schema som beskrivits i avsnitt 6.10 gick det att se en plockförbättring på i genomsnitt 5-10% för en överväldigande majoritet av dimensionerna man kör på sågverket.

Andelen dubbla brädor blev två gånger fler med modellinställningarna i jämförelse med befintliga receptinställningar. Detta är såklart inte önskvärt och förklaras ytterligare i nästa kapitel.

8 Diskussion

8.1 Varför tillämpa maskininlärning?

Oavsett om man finner att maskininlärningen leder till processförbättringar eller inte får man oundvikligen många ovärderliga insikter i sin process under modellutvecklingens gång. För att kunna behandla datan väl tvingas man till gedigna statistiska analyser. Den förbättrade processförståelsen är inte enbart användbar i modellbyggningssyfte utan kan också vara användbar vid framtida maskinutveckling.

En intressant fördel med maskininlärning är att man inte behöver identifiera några samband själv. Som modellkonstruktör skapar man förutsättningarna för att en maskininlärningsmodell ska kunna byggas genom smart databehandling och modellval. Sedan utför modellen det svårare arbetet och identifierar relevanta samband.

8.2 Vad kan sägas om resultatet?

Det första som är viktigt att poängteras är att modellen inte lyckades minska andelen dubbla bitar. Detta kan bero på flera saker. Dels fanns det inte lika mycket data som kan användas för att förutspå RDB. Det kan också vara så att maskinen i sitt nuvarande skick inte kan plocka fler bitar utan att samtidigt öka andelen dubbla brädor.

Resultatet som visar att maskinen plockar 5-10% bättre i jämförelse med receptinställningarna är mer betydande, speciellt eftersom förbättringen utfördes helt utan några mekaniska förändringar på maskinen. Det innebär att arbetet visat att maskinen kan köras effektivare om man är beredd att hantera fler dubbla bitar.

8.3 Förbättringar i framtiden

8.3.1 Predict AND optimize

I detta arbete har konstruktionen av prediktionsmodellen och optimeringsmodellen i stora drag tagits fram separat. Det finns emellertid en artikel [14] som utforskar en metod där prediktionsmodellen konstrueras med det huvudsakliga optimeringssyftet i åtanke. Istället för "predict-then-optimize" sker alltså "predict-and-optimize". Detta kan potentiellt leda till bättre optimeringar.

8.3.2 Kvalitet

Modellen i sin nuvarande skepnad tar enbart hänsyn till virkets dimension. Det finns däremot en parameter till som skulle kunna ha stor betydelse: virkeskvalitet. Under arbetets gång valdes virkesdimensioner ut som viktigast ur optimeringssynpunkt. Kvalitet kan också ha en viss betydelse, men det fanns inte nog med tid att samla in bra dataunderlag på olika kvaliteter för att säga något om dess påverkan.

I ett framtida arbete bör följande beaktas rörande kvalitet:

- Fundera på hur kvalitet bäst kan inkluderas som input i en modell. De primära alternativen är **OneHotEncoder** eller att översätta varje kvalitet till ett numerisk värde som beskriver hur väl kvaliteten betar sig i processen.
- Om en **OneHotEncoder** används kan det vara lämpligt att konstruera två modeller. En modell som är extra bra på att optimera de dimensioner som det finns mycket data på, och en modell som ger ett good-enough resultat överlag - även på kombinationer där specifik data saknas.
- Om kvaliteten översätts till ett numeriskt värde är det viktigt att detta görs på ett för modellen användbart sätt. Ett högre värde skulle exempelvis kunna innebära att brädorna är bättre ur processens synpunkt - att de inte fastnar ovanpå varandra lika lätt.

8.3.3 Dynamisk modell

I detta arbete används en statisk modell. Ingen hänsyn tas till de många dynamiska effekter som sker då elevatorprocessen körs. Med en dynamisk modell som hela tiden läser av momentan maskindata och justerar parametrarna allteftersom skulle det gå att ta hänsyn till dynamik på ett helt annat sätt. För att detta ska vara genomförbart måste processen fysiskt uppgraderas. Ställskenorna måste gå att reglera mer exakt och snabbare. En mer exakt bakåttstegningsalgoritm måste också implementeras för att komma åt det relevanta upplockningsögonblicket för varje plank.

För närvarande används laserdistansmätare för att mäta nivån i diken. En mätare används per dike. Om man hade mätt på fler ställen samtidigt och tagit ett medelvärde hade det potentiellt kunnat leda till mer stabila och pålitliga dikesnivåer, vilket skulle kunna vara användbart för en dynamisk modell.

8.3.4 Elevator 1

I arbetet har fokus enbart legat på Elevator 2 eftersom det är ytterst här som resultatet produceras. Elevator 1 har däremot också en påverkan på resultatet, då den matar dike 2 med virke. En bättre reglering av Elevator 1 skulle potentiellt kunna skapa en jämnare nivå i dike 2.

8.3.5 Inför en kostnadsfunktion

Optimeringen i detta arbete kasserar alla parametervärden med en RDB som ligger ovanför ett önskat gränsvärde. Sedan väljs den kombination med bäst BPM ut som den bästa kandidaten. Den behöver däremot inte vara den bästa kandidaten om man också tar hänsyn till RDB. Om modellen visar att en inställning leder till 191 BPM och 0,8% RDB, och att en annan inställning leder till 190 BPM och 0,1% RDB, kommer modellen att välja ut den första. Detta trots att vinsten överlag troligtvis är större för det andra alternativet. Man

skulle därför kunna implementera någon typ av kostnadsfunktion som finner den verkligt bästa parameterkombinationen.

8.3.6 Annan BPM-filtrering i träningsstadiet

Mot slutet av arbetet i avsnitt 6.10 implementerades en kraftigare filtrering av BPM i syfte att skala bort dynamiska effekter och få en mer rättvis bild av BPM. Detta gjordes enbart för resultatmätning. Det är inte otänkbart att om denna filtrering utfördes *innan* träningen av maskininlärningsmodellen, att modellens prestanda skulle förbättras.

8.3.7 Ta hänsyn till dåliga parametrar

För närvarande tillämpas filtrering av BPM-data som hamnar en viss procent-sats under medianen. Risken med denna typ av filtrering är att viktig data filtreras bort. Om en parameterkombination resulterar i extremt dålig BPM kommer det paketet inte tas med i datan trots att det borde göra det. Syftet med denna filtrering är att ta bort så många dynamiska effekter som möjligt. Men för närvarande sker det på bekostnad av minskad mängd relevant process-data. Detta skulle potentiellt kunna lösas med filtreringsschemat som beskrivs i avsnitt 6.10 innan en modell hade tränats.

Ett annat problem som finns i nuläget är att under insamlandet av slumpdata stängde operatörerna av de slumpade parametrarna om maskinen presterade för dåligt. Detta leder naturligt till att det finns färre data på när modellen presterat dåligt. Modellen har således mer information om när maskinen presterar bra, eller åtminstone mediokert, men ingen information om när den presterat riktigt dåligt. Detta är ett problem eftersom modellen inte får feedback över vilka parameterkombinationer som ska undvikas.

8.3.8 Psykologiska förbättringar

Operatörerna som arbetar i sågverket har möjlighet att stänga av hela eller delar av modell om de inte anser att dess rekommendationer är bra. Det finns inlärd tankar kring vad som anses vara "bra" inställningar. Även om modellen i praktiken uppfyller kriterierna som anges finns en mänsklig benägenhet att minska ens arbetsbörda genom att välja inställningar som resulterar i färre dubbla brädor.

Ett förslag på hur detta skulle kunna lösas är att visa operatörerna i realtid vilken förändring deras manuellt inställda parametrar skulle ha på BPM och RDB. Om operatörerna då skulle se effektivitetstappet alternativa inställningar leder till kan de bli mer medvetna om vad inställningarna får för effekt i praktiken.

9 Slutsatser

- Det är fullt möjligt att låta en maskininlärningsmodell optimera parametrar som styr processen trots att datan inte nödvändigtvis innehåller optimalt inställda maskininställningar. Detta görs genom att en maskininlärningsmodell används för att förutspå verkligheten, varefter ett optimeringsskal utvärderar modellen upprepat tills optimala inställningar identifierats.
- En maskininlärningsmodell kan förbättra prestandan av befintliga, äldre industriella maskiner trots att dessa inte är ideala i maskininlärnings-sammanhang.
- Oavsett om bygget av en maskininlärningsmodell mynnar ut i en prestandaförbättring eller inte kan sekundära vinster i form av ökad processförståelse och kvantitativ dataanalys räcka för att rättfärdiga tillämpningen av maskininläring.
- Den del som tog störst del av tiden i utvecklandet av en maskininlärningsmodell är att bearbeta datan för att få så mycket utav en modell som möjligt.
- Prestandamätningar måste utformas så att irrelevanta effekter filtreras bort. I utformandet av sågverksmodellen var det relevant att filtrera bort dynamiska BPM-effekter.
- Vid utformning av olika typer av datatransformationer är det en bra princip att börja med enkla scheman först och sedan utvärdera ifall mer sofistikerad filtrering hade varit fördelaktig.

Referenser

- [1] *About pandas*. Inhämtat 29/4 2025. URL: <https://pandas.pydata.org/about/>.
- [2] *Beckhoff Information System - T_FILETIME64*. Inhämtat 18/4 2025. URL: https://infosys.beckhoff.com/english.php?content=../content/1033/tcplclib_tc2_utilities/10462547339.html&id=.
- [3] Totalförsvarets forskningsinstitut. *Glimt - vårt nya vapen*. Inhämtat 2/5 2025. URL: <https://www.foi.se/nyheter-och-press/nyheter/2025-02-04-kollektiv-intelligens-ger-glimt-av-framtiden.html>.
- [4] Aurelien Geron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2nd. Sebastopol, CA, USA: O'Reilly Media, Inc., 2019. ISBN: 978-1-4920-3264-9.
- [5] Aurelien Geron. *Machine Learning Notebooks, 3rd edition*. GitHub. 2022. URL: <https://github.com/ageron/handson-ml3>.
- [6] *ISO 8601 - Date and time format*. Inhämtat 18/4 2025. Mars 2020. URL: <https://www.iso.org/iso-8601-date-and-time-format.html>.
- [7] Joseph E. Jabour, Althea C. Henslee, Haley R. Dozier, Indu Shukla, Brandon Hansen, Abderahim Salhi och Ian Dettwiller. "Automated Machine Learning Model Selection Analysis". I: *2023 Congress in Computer Science, Computer Engineering, and Applied Computing (CSCE)*. Las Vegas, NV, USA, 2023, s. 2702–2704. DOI: 10.1109/CSCE60160.2023.00435.
- [8] Gyeongho Kim, Soyeon Park, Jae Gyeong Choi, Sang Min Yang, Hyung Wook Park och Sunghoon Lim (2024) "Developing a data-driven system for grinding process parameter optimization using machine learning and metaheuristic algorithms". *CIRP Journal of Manufacturing Science and Technology*, Volym 51: s. 20–35: ISSN: 1755-5817: URL: <https://www.sciencedirect.com/science/article/pii/S1755581724000518>.
- [9] *Matplotlib home page*. Inhämtat 29/4 2025. URL: <https://matplotlib.org/stable/>.
- [10] Philippe Mongrain, Nadjim Fréchet, Brian Thompson Collart och Yannick Dufresne (2025) "Working the Crowd: Citizen Forecasting, Sophistication and Diversity in Canadian Federal and Provincial Elections". *Canadian Journal of Political Science*, Volym 58 1: s. 68–94: URL: <https://www.cambridge.org/core/journals/canadian-journal-of-political-science-revue-canadienne-de-science-politique/article/working-the-crowd-citizen-forecasting-sophistication-and-diversity-in-canadian-federal-and-provincial-elections/72A94648A42175DAD72DA93D41AA563E>.
- [11] *NumPy home page*. Inhämtat 29/4 2025. URL: <https://numpy.org/>.

- [12] *scikit-learn documentation*. Inhämtat 29/4 2025. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html>.
- [13] Cristina Tavares, Nathalia Nascimento, Paulo Alencar och Donald Cowan (2025) "Variability-Aware Machine Learning Model Selection: Feature Modeling, Instantiation, and Experimental Case Study". *IEEE Access*, Volym 13: s. 62527–62542: URL: <https://ieeexplore.ieee.org/document/10950398>.
- [14] Toon Vanderschueren, Tim Verdonck, Bart Baesens och Wouter Verbeke (2022) "Predict-then-optimize or predict-and-optimize? An empirical evaluation of cost-sensitive learning strategies". *Information Sciences*, Volym 594: s. 400–415: ISSN: 0020-0255: URL: <https://www.sciencedirect.com/science/article/pii/S0020025522001542>.
- [15] Li-Chih Wang, Chun-Chih Chen och Chen-Chi Hsu (2022) "Applying machine learning and GA for process parameter optimization in car steering wheel manufacturing". *The International Journal of Advanced Manufacturing Technology*, Volym 122 11–12: s. 4389–4403.