Utvärdering och tillämpning av Automatiserade maskininlärningsramverk för finansiell dataanalys

Examensarbete inom högskoleingenjörsprogrammet datateknik

Emil Johansson

Oscar Lundgren

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK
CHALMERS TEKNISKA HÖGSKOLA

Göteborg, Sverige 2025
www.chalmers.se

Examensarbete 2025

Utvärdering och tillämpning av Automatiserade maskininlärningsramverk

Automatiserad maskininlärning för finansiell dataanalys

Emil Johansson
Oscar Lundgren

Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg 2025

Utvärdering och tillämpning av Automatiserade maskininlärningsramverk för finansiell -dataanalys
Automatiserad maskininlärning för finansiell dataanalys

Handledare: Sakib Sistek
Examinator: Nicholas Smallbone
Företag: Himode AB

Examensarbete 2025
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Institutionen för data- och informationsteknik
Göteborg 2025

Utvärdering och tillämpning av Automatiserade maskininlärningsramverk för finansiell -dataanalys
Automatiserad maskininlärning för finansiell dataanalys
EMIL JOHANSSON
OSCAR LUNDGREN
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola

Sammanfattning

Maskininlärning används idag av många olika sektorer med målet att förutsäga framtiden med hjälp av tekniska databaserade prognoser. Finanssektorn har under en lång period varit ett område som utvecklat teknik för att nyttja möjligheten att prediktera utfall för aktiekurser eller handelsbeslut. Under utvecklingens gång har denna typ av maskininlärning även övergått till det automatiserade med hjälp av automatiserade maskininlärningsramverk. Detta projekt syftar på att tillämpa dessa ramverk och sedan utvärdera resultaten för diverse förutbestämda faktorer.

En stor samling ramverk utvärderades under projektets testfas där de mest lämpade följde med för djupare analys och tillämpning. Faktorer som användarvänlighet, komplexitet och prestanda noterades och de kvarlevande ramverken testades med verklig data. Kod för datamärkning med regressions- och klassifikationsuppgifter utvecklades för ramverken i förberedelse för utfallstester.

Resultaten från utfallstesterna noterades för varje individuellt ramverk och presenterades med hjälp av tabeller och grafer. Hädanefter jämfördes ramverkens resultat för varje förutbestämd faktor och därefter redovisades ramverkens lämplighet inom respektive kategori samt i dess helhet.

Abstract

Machine learning has in the current age expanded to be used by many different sectors with the purpose of predicting the future using technical data-based forecasts. The financial sector has for a long time been a field that has led the development of this technology in the hopes of predicting outcomes for stock prices or trading decisions. In the course of its development, this type of machine learning has also moved into the automated realm by using automated machine learning frameworks. This project aims to apply these frameworks and then evaluate the results based on various predetermined factors.

A large collection of frameworks was evaluated during the testing phase of the project, the most suitable ones were then used for deeper analysis and implementation. Factors such as ease of use, complexity and performance were noted and the remaining frameworks were tested with real data. Data labeling code with regression and classification tasks was developed for the frameworks in preparation for backtesting.

The results of the backtests were logged for each individual framework and presented using tables and graphs. The backtesting results of the frameworks were then compared for each predetermined factor and the suitability of the frameworks within each category and as a whole was reported and presented.

Akronymer

AutoML	Automatiserad maskininlärning
HPO	Hyperparameter optimering
XGBoost	Extreme Gradient Boosting
LGBM	Light Gradient Boosting Machine
OHLC	Open - High - Low - Close
FLAML	Fast and lightweight AutoML library

Innehållsförteckning

1. Introduktion

Finansiell dataanalys handlar om att tolka och dra slutsatser från stora och varierande datamängder vilket sedan kan användas som grund för riskhantering, investeringsstrategier samt ekonomiska beslut. Denna data inkluderar olika former av information, exempelvis pris och volym för diverse finansiella instrument samt mer fundamentala nyckeltal som intäkter och skulder. Detta visar i större utsträckning företagets ekonomiska hälsa, men även textbaserad information som analytikerrapporter, nyhetsartiklar samt sociala medier har stor påverkan på det övervägande sentimentet. Sentiment innebär inte endast insikt för företagets framtid utan har även förmågan att direkt påverka marknadsvärdet. För att effektivt kunna bearbeta olika former av finansiell data krävs det kraftfulla verktyg för att förenkla processen, vilket har lett till utvecklingen av en stor mängd unika *automatiserade maskininlärningssramverk*. AutoML har som huvudmål att automatisera olika steg inom maskininlärning för att göra det enklare att använda samt bygga maskininlärningsmodeller. AutoML har därför möjligheten att göra maskininlärning mer tillgängligt och tidssparande genom att automatisera många av de komplexa och tekniska stegen som krävs för att konstruera, träna och implementera maskininlärningsmodeller. Utvecklingen av AutoML-ramverk har under senare tid accelererats och antalet ramverk har ökat markant. Detta har lett till ett brett ekosystem av ramverk där det kan vara svårt för bolag och utvecklare att bestämma vilket ramverk som bäst är anpassat för sina egna syften och mål.

1.1 Syfte

Syftet med projektet är att utvärdera hur AutoML-ramverk kan användas för att stödja ett tekniskt aktiebolag under utvecklingsfasen, innan de har påbörjat en aktiv investering. Bolaget arbetar under detta skede med utveckling relaterat till vårt arbete, såsom datainsamling i form av dataskrapning och analys av sentiment med mera. Vårt arbete syftar på att ge underlag för vilka verktyg som lämpar sig bäst för företagets framtida tillämpning av kvantitativ modellering och datadrivna investeringsstrategier.

1.2 Mål

Huvudmålet med vårt projekt är inte enbart att undersöka hur effektiva existerande AutoML-ramverk är på att utvärdera utfall utifrån finansiell data utan handlar även om att undersöka och utvärdera olika faktorer relaterade till ramverkens användning. De viktigaste faktorerna för detta projekt inkluderar: användarvänlighet, prestanda, komplexitet samt körtiden för ramverken i förhållande till resultaten. Det är även nödvändigt för ramverken att kunna hantera diverse datatyper såsom textdata och tidsserier.

1.3 Avgränsningar

Utvecklingen kommer att avgränsa sig från att vara helt resultatfokuserad. Målet för detta projekt är inte att skapa en kvantitativ miljardvinnande strategi som hävdar sig resultera i det bästa utfallet för direktinvestering. Resultatvärden som precision skall inte vara det enda huvudfokuset eftersom arbetet handlar om AutoML-ramverken för sig och inte enbart resultaten producerade av koden vi har framhävt för varje ramverk. Under projektets första fas kommer testerna att begränsas till tio etablerade AutoML-ramverk. Dessa ramverk kommer att utvärderas i koppling till de förutbestämda faktorerna och sedan begränsas ytterligare till de fem mest lämpade ramverken. Valet av tio ramverk gjordes för att säkerställa att arbetet kunde utföras under den bestämda projektperioden.

2. Metod

2.1 Programmeringsmiljö

Python valdes som det primära kodspråket för detta arbete på grund av dess flexibilitet och breda bibliotek. Jupyter notebook användes som arbetsyta för att enkelt kunna dela upp och samla dokumenten med kod för respektive AutoML-ramverk. Plattformen Anaconda utnyttjades för installation av de korrekta biblioteken samt för dess användarvänlighet.

2.2 Val av AutoML-ramverk

Under arbetet har endast ramverk med öppen källkod testats. Det finns en stor mängd av dessa och på grund av projektets omfattning och tidsbegränsning så valdes ett antal av dessa för vidare testning. Ramverken blev utvalda efter deras kapacitet, lämplighet och stöd från användare. Mot slutet av projektet hamnade det största fokuset på ett fåtal av dessa ramverk som uppnådde kraven på bästa möjliga sätt, vilka var H2O, AutoGluon och FLAML.

2.3 Planeringsrapport

Under de första veckorna av projektet skapades en planeringsrapport som var menat att säkerställa projektets mål och avgränsningar. Planeringsrapporten inkluderade även ett tidsschema, vilket inte skapades för att producera tydliga delmål eller exakta resultat eftersom det hade varit omöjligt att säkerställa vid projektets start. Schemat var istället avsett för att skapa en tydlig struktur av arbetets uppdelning under projektets livslängd.

2.4 Datainsamling

Under projektets gång användes olika former av data som insamlats på varierande vis. Först och främst användes simpla dataset från kaggle [1] vilka inkluderade gammal aktiedata från Nasdaq vilket till stor del bestod av *OHLC* data, där varje akties tidsperiod representeras med fyra värden på priset vilka är dess öppningskurs (O), högsta kurs (H), lägsta kurs (L) och stängningskurs (C). Under senare skede användes mer aktuell data från Nasdaq, försett av handledaren på bolaget. Datan var separerad baserat på period, vilket innebar tillgång till dagsdata, timdata och även tickdata. Datan inkluderade även aktier från bolag som har gått i konkurs vilket är relevant för att undvika överlevnadsbias.

2.4.1 Variabelteknik

Eftersom dataset inte alltid har de mest optimala variablerna så är det vanligt att lägga till ytterligare av dessa för att få bättre resultat. Aktiedata innehåller oftast endast OHLC samt volym som variabler, vilka inte är tillräckliga för modellerna att kunna göra extraordinära prediktioner utifrån. Därav användes en del tekniska indikatorer till hjälp som exempelvis relativt styrkeindex, glidande medelvärde med olika dagars intervall och genomsnittligt verkligt intervall. En del AutoML-ramverk lägger till variabler på egen hand, men för att få en rättvis jämförelse mellan dessa så lades variablerna till manuellt på dataseten.

Det säkerställdes också att varje aktie sorterades kronologiskt efter datum för att beräkningarna av tekniska indikatorer och segmentering av data utgick från ett korrekt historiskt perspektiv. Eftersom beräkningar av glidande medelvärden med exempelvis trettio dagars intervall skapar initiala luckor, så kallat *inte-ett-nummer*, behövdes dessa fyllas i. För detta fylldes värdena i bakåt i tiden för de första trettio dagarna för att behålla kontinuiteten i tidsserien och för att vissa ramverk inte tillåter värden utan något innehåll.

2.5 Utvärderingsmetoder

Under projektets gång testades en rad olika AutoML-ramverk på både uppgifter av typen regression samt klassifikation. För dessa uppgifter valdes de vanligaste och mest informativa måtten ut för att jämföra ramverken på ett rättvist sätt.

För uppgifterna av typen regression valdes *rotmedelskvadratfelet* och *genomsnittligt absolutfel* som mått. Rotmedelskvadratfelet anger roten ur medelkvadratfelet och visar större känslighet kring större avvikelser medan det genomsnittliga absolutfelet beskriver den genomsnittliga felmarginalen i samma enhet som priset [2]. Lägre värde visar på att modellen kunnat prediktera bättre än ett högre värde.

För uppgifterna av typen klassifikation används *precision*, *återkallning*, *f1-poäng* och *noggrannhet*. Precisionen visar andelen av de instanser där modellen predikterat som positiva som faktiskt är positiva, hög precision innebär därav få falska alarm och lågt värde indikerar att modellen ofta gör fel [3]. Återkallning anger andelen av de verkliga positiva instanserna som modellen lyckas hitta. Hög återkallning innebär därför att modellen lyckas hitta de flesta av rätt instanser medan lågt värde visar på att många verkliga positiva instanser missas. Om precisionen för köpklassen är låg kan alltså modellen generera många falska köpsignaler vilket kan leda till förlust av kapital. Är istället återkallningen låg för exempelvis köpklassen kommer modellen missa många av de verkliga köptillfällena.

F1-poäng används som en balans av medelvärdet mellan precision och återkallning, eftersom dessa två ofta står i konflikt mot varandra. Genom att använda sig av f1-poäng får man en mer nyanserad bild av modellens styrkor och svagheter. Noggrannhet är ett mått vilket visar andelen instanser som klassificerats korrekt gentemot det totala antalet instanser.

3. Teknisk bakgrund

3.1 Översikt av AutoML

Automatisk maskininlärning är processen av att automatisera uppgifter för att tillämpa maskininlärning på verkliga problem. Det är alltså kombinationen mellan automation och maskininlärning. Den höga graden av automation i AutoML syftar till att tillåta icke-experter till att använda maskininlärningsmodeller och tekniker utan att det kräver att de blir experter inom området. AutoML ger användaren möjligheten att testa mycket fler modeller under en kortare period för att hitta bättre lösningar utan att behöva koda varje steg manuellt. AutoML integrerar en stor mängd olika tekniker, till exempel regressionsmodeller, HPO (optimering av hyperparametrar) och mycket mer.

3.1.1 Optimering av hyperparametrar

Varje maskininlärnings algoritm har en uppsättning av *hyperparametrar*, vilka är inställningar som definieras innan träning kan påbörjas. Dessa kan bestå av exempelvis inlärningshastighet, maxdjup på beslutsträd eller antal noder per lager i ett neuralt nätverk [4]. För att varje modell ska nå sin maximala prestanda behövs dessa parametrar optimeras, där AutoML-ramverk använder sig av en del olika metoder. Följande tre metoder är några av de mest grundläggande och förekommande inom området.

3.1.1.1 Rutnätsökning

Rutnätsökning fungerar genom att systematiskt söka igenom ett definierat område av tänkbara värden för varje hyperparameter [5]. Modellen tränas och utvärderas sedan på varje möjlig kombination från sökområdet med hjälp av ett prestandamått, oftast korsvalidering på träningsdata. Kombinationen som ger bäst valideringsresultat väljs då som inställning innan modellen används på ny data. Fördelen med denna metod är att den säkerställer att alla fördefinierade alternativ testas och missar därmed aldrig en tänkbar konfiguration. Dock medför detta mer kostnad i form av beräkningar, vilket kan växa fort med antalet hyperparametrar och värden.

3.1.1.2 Slumpmässig sökning

Slumpmässig sökning är en optimeringsmetod där man istället för att systematiskt testa kombinationer, drar ett slumpvis förutbestämt antal kombinationer av hyperparametrar från ett definierat sökområde [5]. För varje kombination tränas och utvärderas modellen, där den bästa uppsättningen behålls. De flesta hyperparametrar har varierande betydelse för modellens prestanda, vilket gör att slumpmässig sökning kan hitta bra inställningar utan att behöva söka igenom hela sökområdet. Detta gör den mer effektiv än rutnätsökning till kostnad av prestanda, eftersom den inte testar alla möjliga kombinationer.

3.1.1.3 Bayesiansk optimering

Bayesiansk optimering är en metod som skapar en enklare modell för att fånga upp sambandet mellan hyperparametrar och den uppmätta modellprestandan [6]. Den börjar med att testa en del slumpmässiga kombinationer för att skapa en uppfattning om vilka områden av parametrarna som kan vara lovande. Utifrån detta förutspås hur bra prestanda de olika parametrarna kan ge. Sedan används detta för att välja nästa punkt där osäkerheten kring parametrarna är hög eller där förbättringar förväntas bli störst. Iterativt ger varje nytt test ny information och efterhand lär den sig snabbt att hitta de hyperparametrar som ger bäst modellprestanda. Genom detta behöver bayesiansk optimering vanligtvis färre utvärderingar än både rutnät och slumpmässig sökning, speciellt i komplexa sökområden.

3.1.2 Arbetsflöde hos AutoML-ramverk

De flesta AutoML-ramverken följer i regel ett strukturerat arbetsflöde som syftar till att automatisera de centrala stegen inom maskininlärning. Processen inleds med att användaren anger sitt dataset som därefter förbehandlas i form av att saknade värden hanteras och variabler normaliseras så att informationen är i rätt format. Efter detta brukar variabelteknik användas, där de mest relevanta variablerna identifieras och överflödiga tas bort. Detta är avgörande för modellernas förmåga att dra meningsfulla slutsatser från data.

Därefter väljer ramverken ut lämpliga maskininlärningsmodeller genom att testa ett brett urval av algoritmer. När en lovande uppsättning modeller har hittats, påbörjas optimering av hyperparametrar på respektive modell. HPO handlar om att hitta de bästa parametrarna för en modell innan den tränas. AutoML-ramverk använder olika automatiserade metoder för detta som exempelvis rutnätsökning eller bayesiansk optimering.

När modellerna tränats och optimerats utsätts de för en noggrann utvärdering. Ramverken säkerställer även att resultaten är generaliserbara och inte överanpassade på träningsdatan genom tillämpning av metoder som korsvalidering. Sedan väljs den modell som presterat bäst enligt definierade mått som exempelvis noggrannhet, precision eller återkallning.

Till slut genererar AutoML-ramverken den bästa modellen enligt resultaten samt ibland en rapport över alla modeller som har testats i form av en ledartavla. Ledartavlan syftar till att visa prestandan i olika mått för respektive modell och de viktigaste faktorerna som påverkat resultaten.

3.2 Maskininlärningsalgoritmer och -modeller

I maskininlärning finns det olika typer av algoritmer och modeller som AutoML-ramverk använder sig av. Dessa finns i olika typer som linjära, trädbaserade, neurala nätverk och ensemble-metoder.

3.2.1 Linjära modeller

Linjära modeller antar att sambandet mellan ingångsvariabler och målvariabeln kan beskrivas som en linjär kombination. Tack vare sin simplicitet så är de snabba på att träna och lättare att tolka men saknar förmågan att fånga icke-linjära mönster.

3.2.1.1 Linjär regression

Linjär regression är en metod som fungerar genom att anpassa en linjär ekvation till en datamängd för att på så sätt beskriva sambandet mellan en beroende variabel och en eller flera oberoende variabler [7]. Genom att använda sig av exempel där variablerna och det verkliga utfallet är kända kan metoden pröva olika varianter på linjens lutning för att minimera avståndet mellan linjen och de faktiska punkterna. För att göra detta beräknas först medelvärdet av de kvadrerade felen. Felet är skillnaden mellan de faktiska värdena och vad linjen förutspår. Därefter kan parametrarna justeras stegvis för att minimera felet.

3.2.1.2 Logistisk regression

Logistisk regression är en statistisk metod vilken används för att förutsäga sannolikheten för att en specifik händelse inträffar. Likt linjär regression fungerar metoden genom att först beräkna en linjär kombination av insatsvariablerna men istället för att måla upp utfallet som en rak linje, appliceras en logistisk funktion på resultatet [8]. Detta omvandlar sedan värdena till ett intervall mellan noll och ett, vilket tolkas som sannolikheten att den beroende variabeln är ett, alltså att händelsen inträffar.

3.2.2 Trädbaserade algoritmer

Trädbaserade algoritmer delar upp datan stegvis efter villkor på olika variabler, vilket skapar en trädlik struktur. De kan hantera både enkla och mer avancerade samband utan att behöva ändra data.

3.2.2.1 Gradient boosting

Gradient boosting är en metod menat för att successivt förbättra modeller genom att rätta till fel. Metoden bygger flera enkla svaga modeller där varje modell fokuserar på felen av den tidigare modellen [9]. Genom små förändringar bygger Gradient Boosting upp en modell av enkla prediktorer, oftast beslutsträd, där varje nytt träd korrigerar de fel som återstår av tidigare träd. Metoden börjar med att välja en prediktion som startpunkt, exempelvis medelvärdet av målvariabeln, och räknar ut hur fel gissningen är för varje värde. Sedan tränas ett nytt träd för att rätta till felaktigheterna som uppstått från startpunkten. Detta fortgår tills processen upprepats ett antal gånger eller tills förbättringen i förlust-värdet är försumbar. Till slut har man en modell som är summan av alla dessa träd, som har hjälpt till med att förbättra precisionen. Kombinationen av små förbättringar och kontroll av varje träds komplexitet gör att gradient boosting kan fånga upp både linjära och icke-linjära samband, utan förhandskunskap om datasetet.

3.2.2.2 Extreme gradient boosting

Extreme gradient boosting, också kallat *XGBoost,* är ett kraftfullt bibliotek med öppen källkod som var ett av de första med att optimera gradient boosting. Istället för att bygga ett stort beslutsträd kombinerar XGBoost många mindre träd i en sekventiell process, där varje nytt träd har i uppgift att rätta de felen som uppstått från tidigare träd [10]. Algoritmen kan därav finslipa sina förutsägelser och har oftast mycket hög precision.

Algoritmen är skapad för att vara snabb och effektiv. Genom att dela upp data i flera delar som arbetas parallellt tillsammans med histogram-baserade beräkningar för att minimera minnesförbrukning, kan det hantera stora mängder data på ett produktivt sätt. XGBoost har också inbyggda processer för att hantera saknade värden utan någon förbehandling av data och regularisering som hjälper till att undvika överanpassning.

3.2.2.3 Light Gradient Boosting Machine

Light gradient boosting machine, också kallat *LBGM,* är byggt för att producera robusta modeller genom gradient boosting. Likt XGBoost, fungerar det med att modellen successivt lägger till enkla beslutsträd, där varje nytt träd tränas för att rätta till de fel som uppstått i tidigare träd [11]. Till skillnad från andra implementationer så delar dock LGBM inte upp trädens nivå för nivå utan väljer istället att dela den nod som ger störst fel reduktion först. Detta sätt gör att träden blir djupare där det gör mest nytta, vilket leder till snabbare inlärning och mindre fel.

För att hålla nere minnesförbrukning använder sig LGBM av en metod som delar in kontinuerliga variabler i ett antal grupper innan träden byggs på. På så sätt kan algoritmen samla flera datapunkter i varje grupp och därigenom kraftigt minska antalet beräkningar. Dessutom hanterar LGBM kategoriska variabler direkt utan att användaren behöver göra om dessa till så kallade dummy-kolumner. Dessa optimeringar ger en snabb och skalbar modell som kan hantera extremt stora mängder data med hög precision.

3.2.2.4 CatBoost

CatBoost är utvecklat för att kunna hantera *kategoriska variabler* på ett effektivt sätt, vilket är variabler som delas in i klasser eller grupper istället för numeriska värden. Algoritmen är implementerad så att användaren inte behöver omvandla text- eller kategorikolumner till dummyvariabler, alltså från text till binärt [12]. Precis som i tidigare nämnda gradient boosting algoritmer så bygger CatBoost upp en sekvens av beslutsträd där varje nytt träd tränas för att rätta till felen från tidigare träd.

Algoritmen är optimerad för både stabila resultat och snabb körning, vilket innebär att användare enkelt kan använda standardinställningar och få prestanda på bra nivå jämfört med andra bibliotek. Det har även inbyggt stöd för parallell bearbetning, saknade värden och GPU-acceleration vilket gör att det klarar av stora datamängder samtidigt som det undviker överanpassning.

3.2.3 Neurala nätverk

Neurala nätverk kan beskrivas som lager av sammankopplade noder som beräknar viktade summor samt följt av aktiveringsfunktioner. De är speciellt bra på att se komplexa mönster, dock till en högre beräkningskostnad.

3.2.3.1 Flerlager-perceptron

Flerlager-perceptron är ett neuralt nätverk med flera lager av sammankopplade noder, där varje lager bearbetar information för att analysera och modellera samband i datamängden [13]. Dessa lager består oftast av en kombination av ett inmatningslager ett utgångslager och ett eller flera dolda lager. Noderna tar emot en viktad summa av signalerna från tidigare lager och producerar sedan nodernas utgångsvärde med hjälp av en aktiveringsfunktion. Hädanefter justeras viktningen med hjälp av olika funktioner för att förutsägelsen närmare ska efterlikna det verkliga värdet. Flerlager-perceptron kan exempelvis användas för både klassificering och regression och används ofta för mer krävande problem.

3.2.3.2 Återkommande neurala nätverk

Återkommande neurala nätverk där utdatan från tidigare steg används som indata till det verkliga steget med hjälp av ett dolt lager som spårar datasekvenser [14]. Nätverket innehåller alltså sin egen inbyggda minnesfunktion och används ofta när data är sekventiell och vikterna måste hanteras på samma sätt i alla lager, t.ex. i tidsseriedata.

3.2.4 Ensemble

Ensemble-modeller siktar på att öka precision och stabilitet genom att kombinera flera olika basmodeller. Eftersom olika modeller har olika styrkor och svagheter så kan man genom att kombinera dem minska varians och bias. Ensemble Modeller inkluderar tre huvudtyper: *Bagging*, *Boosting* och *Stacking*.

3.2.4.1 Bagging

Bagging används för att reducera varians i modellen genom att parallellt träna flera versioner av exakt samma modell på olika slumpmässiga delmängder av träningsdata [15].

3.2.4.2 Boosting

Under Boosting tränas modeller sekventiellt med målet att få en slutgiltigt viktad kombination av alla modellerna [15]. Detta uppnås genom att varje modell lägger större vikt på misslyckanden av de tidigare modellerna.

3.2.4.3 Stacking

Stacking delar upp problemet från uppgiften till diverse modeller och tillåter dem att göra sin egen prediktion, därefter blir en enskild modell tränad på kombinationen av modellernas gissning med målet att skapa en starkare och mer robust prediktion [15].

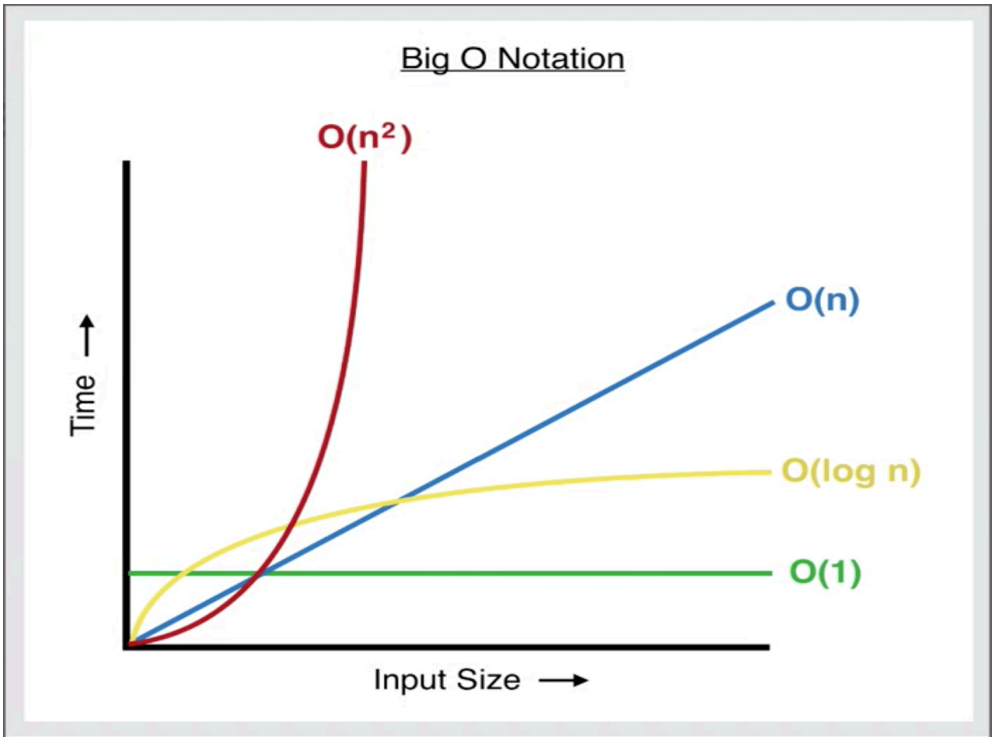
3.2.5 Tidskomplexitet med fler variabler

Tidskomplexiteten hos maskininlärningsmodeller påverkas av antalet variabler i ett dataset och varierar beroende på vilken typ av modell som används. Dessa kan beskrivas i form av *ordonotation* (Big O notation), som används inom datavetenskap och matematik för att ge mått på hur tung en algoritm är beräkningsmässigt [16].

Linjära modeller, som linjär regression och logistisk regression, har en tidskomplexitet som kan beskrivas med O(1) till O(n), i ordonotation enligt figur 3.1. Det betyder att beräkningstiden ökar i takt med antalet variabler linjärt, eftersom vid varje optimering så måste gradienterna räknas över alla ingångs variabler.

Trädbaserade algoritmer har en annan tidskomplexitet gentemot linjära modeller och påverkas på ett annat sätt av ett ökande antal variabler. I ordonotation kan deras komplexitet beskrivas som O(n^2), med variation beroende på vilken modell som används. Vid varje uppdelningspunkt måste alla variabler testas för att hitta den bästa splitten, vilket innebär att fler variabler kommer öka tiden det tar att bygga varje träd. AutoML-ramverk brukar därav använda sig av exempelvis approximativa metoder och urval av variabler för att minska beräkningskostnaden och förbättra modellens effektivitet.

Neurala nätverk kan hantera stora och komplexa dataset, dock till bekostnad på högre beräkningskostnader. Tidskomplexiteten för dessa kan beskrivas som O(n log n) i ordonotation. När antalet variabler ökar krävs det fler neuroner per lager, vilket kan leda till en exponentiell ökning av beräkningar. För att kunna hantera detta brukar AutoML-ramverk använda sig av exempelvis batch-normalisering vilket kan minska behovet av att hantera irrelevanta variabler och förbättra effektiviteten hos modellerna.



Figur 3.1: Översikt av ordonotation [17]

3.3 Urval av variabler inom AutoML

Urvalet av variabler är en viktig del av processen inom arbetsflödet för AutoML. Genom att identifiera och behålla de mest gynnsamma variablerna kan modeller tränas mer effektivt och samtidigt undvika överanpassning. För detta så använder ramverken en mängd olika tekniker för att automatisera processen kring urvalet av variabler. I detta avsnitt beskrivs tre olika strategier som ofta kombineras i AutoML-ramverk, vilka är filter-, wrapper- och inbäddade metoder.

3.3.1 Filtermetoder

Filter-metoder syftar till att utvärdera varje variabels relation till målvariabeln och är även oberoende av vilken modell som väljs. Som ett första steg brukar varians användas, vilket går ut på att sålla bort variabler som nästintill är konstanta [18]. Dessa bär ingen meningsfull information för en modell och är ofta överflödiga. *Pearson-korrelation* är en metod som går ut på att beräkna korrelationen mellan varje enskild variabel och mål variabeln, samt mellan alla variabler. Om variabeln är under eller över en viss tröskel så tas den bort. Detta kan också användas för att hantera starkt korrelerade variabler, där man behåller en av två variabler som mäter ungefär samma sak för att minska komplexiteten. *Mutual information* är en annan vanlig variant som mäter hur mycket två variabler delar med varandra. Ett lågt gemensamt värde indikerar att en variabel inte bidrar med någon unik information, och kan därav tas bort.

3.3.2 Wrappermetoder

Wrappermetoder testar olika delmängder av variabler i kombination med en grundmodell, därav kan urvalet optimeras baserat på faktisk modellprestanda [18]. *Rekursiv variabeleliminering* är en vanlig metod inom detta som går ut på att träna modellen på alla kvarvarande variabler för att på så sätt se vilken av dem som bidrar minst. Variabeln som har lägst koefficient eller minst betydelse för modellen vid varje iteration plockas bort, sedan upprepas processen tills ett fördefinierat antal variabler finns kvar eller tills prestandan understiger en viss gräns. Wrappermetoder ger ofta bättre resultat än filtermetoder, men har också betydligt dyrare beräkningar.

3.3.3 Inbäddade metoder

Inbäddade metoder integrerar urval av variabler i själva träningsprocessen genom att använda modeller med inbyggt urval eller regularisering [18]. Trädbaserade modeller har inbyggda metoder för att kunna beräkna vikten av varje variabel. AutoML använder sig av dessa vikter för att kunna identifiera och ta bort mindre viktiga variabler för att undvika överanpassning samt minska träningstiden. *Permutationsvikt* är en liknande metod som mäter hur mycket modellens prestanda försämras när värdet på en slumpmässigt vald variabel ändras. En stor försämring visar att variabeln är viktig och bör behållas.

Två vanliga typer av regularisering är *L1* (Lasso) och *L2* (Ridge), som båda inför straff för stora koefficienter för att minska överanpassning [19]. L1-regularisering lägger till ett straff på summan av de absoluta värdena av modellens vikt koefficienter. Detta sätter vissa vikter till noll, vilket innebär att motsvarande variabler kan elimineras. L2-regularisering försöker hålla vikterna små, utan att sätta dem till exakt noll. Här straffas istället summan av vikternas kvadrater, alltså straffas stora vikter mer än små. L2 är särskilt användbart när det finns många variabler som är korrelerade, eftersom vikterna kan fördelas jämnt mellan de korrelerade variablerna. Detta i sin tur minskar risken för att en enskild variabel får för stor påverkan på modellen.

3.4 Beskrivning av utvalda AutoML-ramverk

I denna sektion beskrivs vilka AutoML-ramverk som testats under projektets gång samt hur de fungerar. Syftet är att ge en översikt över varje ramverks huvudfunktioner, styrkor och typiska användningsområden.

3.4.1 Fast and lightweight AutoML library

Fast and lightweight AutoML library, också kallat *FLAML*, är ett Python-bibliotek som är skapat för att hitta bra modeller till regression och klassifikation baserat på användarens data [20]. Ramverket lägger stor vikt på att effektivisera sökandet av modeller med så låg minnesförbrukning som möjligt. Detta görs genom att den iterativt bestämmer inlärningsmodell, hyperparametrar, provstorlek och provtagningsstrategi som ska användas, samtidigt som det väger deras sammansatta påverkan på både beräkningskostnad och fel under sökningens gång. FLAML stödjer också snabb och ekonomisk optimering som adaptiv sampling samt tidig stoppning som används för att balansera modellkvalitet gentemot beräkningskostnad. Det har även stöd för flera av de främsta maskininlärningsalgoritmerna som XGBoost, LGBM och CatBoost.

3.4.2 H2O AutoML

H2O AutoML är en del av plattformen H2O.ai som erbjuder automatiserad maskininlärning. Syftet med ramverket är att effektivisera och förenkla arbetsflödet för att träna samt jämföra modeller inom maskininlärning [21]. Ramverket är designat med ett gränssnitt med få parametrar för att underlätta för användaren, som endast behöver ange sitt dataset och identifiera vilken kolumn som har målvariabeln. Vidare så finns det möjligheter att ange en tidsgräns på hur länge ramverket ska köra, vilka algoritmer som ska köras och hur många modeller som ska tränas. H2O har även inbyggd hyperparametersökning för att automatiskt hitta de mest optimala hyperparametrarna för varje modell.

Efter att de individuella modellerna har tränats så bygger ramverket staplade ensemble-modeller, vilka nästan alltid har högre precision än enskilda modeller. Efter att det blivit klar med träningen av modellerna skrivs en ledartavla ut. Ledartavlan innehåller resultaten från varje modell som gör det enkelt att jämföra dem mot varandra och se vilken modell som presterat bäst.

3.4.3 AutoML Mljar-supervised

Mljar-supervised är ett Python-paket för automatisk maskininlärning som hanterar tabulär data [22]. Ramverket lägger mycket vikt i att visualisera resultaten för användaren. Efter en AutoML-körning så genereras det automatiskt en mapp med en HTML-rapport som innehåller alla testade modeller. Rapporten sparar varje del av arbetsflödet och användaren kan enkelt se hur varje enskild modell presterat med hjälp av diagram över prestandan och vikten av alla variabler. Mljar justerar hyperparametrar automatiskt för att förbättra modellens prestanda genom *not-so-random-search* metoden. Istället för att slumpmässigt testa parametrar, som i slumpmässig sökning, så definieras i förväg en uppsättning tänkbara värden för varje hyperparameter. Sedan så väljs nya kombinationer av värdena vid varje iteration som sparas och används vidare för att begränsa sökningen till de mest lovande parametrarna.

3.4.4 Auto-sklearn

Auto-sklearn är ett AutoML-ramverk som är byggt ovanpå Python-biblioteket scikit-learn [23]. Ramverket söker automatiskt igenom flera olika maskininlärningsalgoritmer och förbehandlingstekniker för att skapa ett optimalt arbetsflöde anpassad efter användarens dataset. För HPO använder ramverket sig av bayesiansk optimering för att effektivt justera hyperparametrarna för de utvalda modellerna. Efter att en uppsättning bra modeller har hittats så skapas en ensemble med dessa för att ytterligare förbättra precisionen och robustheten. Auto-sklearn använder sig också av meta-lärande för att starta processen av optimering. Genom användning av tidigare erfarenheter från liknande dataset, så kan det effektivare hitta konfigurationer som passar bra på användarens dataset.

3.4.5 AutoGluon

AutoGluon-Tabular är ett AutoML-ramverk med öppen källkod som utvecklats av Amazon. Till skillnad från andra ramverk som fokuserar mycket på modellval och hyperparametrar, så lägger AutoGluon mycket vikt vid skapandet av ensembler [24]. Det hanterar även automatiskt vilken typ av problem datan innehåller som klassifikation eller regression, tar hand om saknade värden och justerar hyperparametrar till varje modell som testas. Under en körning börjar AutoGluon med att fylla i saknade värden och omvandla kategorivariabler till tal. Därefter tränas och testas ett antal förinställda modeller och sedan justeras hyperparametrar för dessa. Initialt sorterar ramverket bort de modeller som presterar sämre och fortsätter köra de modeller som verkar lovande samt optimerar dessa med exempelvis bayesiansk optimering. När varje basmodell har finjusterats så skapar AutoGluon en ensemble, alltså en kombination av ett antal av modellerna som tränats. Detta görs genom att använda en meta-modell, som lär sig hur de olika basmodellerna kompletterar varandra där resultatet blir en ensemble som har bättre precision än vad en enskild modell kan uppnå.

3.5 Maskininlärning för finansmarknader

Maskininlärning inom finans handlar i dess grund om att analysera historiska datamängder i ett försök att förutspå marknadsrörelser för att göra gynnande investeringbeslut. För att hitta mönster som en människa har möjligheten att missa matas algoritmer med olika former av data. Allt från OHLC-data till nyhetsartiklar kan användas för att optimera portföljer. För att uppnå detta används olika former av metoder såsom neurala nätverk, beslutsträd och mycket mer.

3.5.1 Tidsserieanalys

En stor del av finansdata är ofta tidsseriedata med exakta tidpunkter och specifika datum. För att identifiera plötsliga förändringar och trender behöver man kunna modellera dessa värden för att generera prognoser som efterliknar det verkliga värdet så bra som möjligt. För att uppnå detta används en rad av olika metoder och tekniker, till exempel tidsfönster, förskjutningar och hyperparametrar.

3.6 Anaconda & Jupyter Notebook

Anaconda är en källkods-distribution av programmeringsspråken Python och R, vilket gör det enkelt för användaren att hantera distribution och pakethantering [25]. Plattformen inkluderar ett stort antal förinstallerade bibliotek samt miljöhanterings-funktioner som skapar ett effektivt arbetsflöde. Jupyter Notebook är en utvecklingsmiljö skapad främst för dataanalys och visualisering för att underlätta datavetenskapliga syften. Notebook använder sig av ett cellbaserat system vilket sparar tid eftersom koden kan delas upp mellan cellerna. Detta innebär att hela programmet inte alltid måste köras från början till slut utan körningen kan istället fördelas inom olika celler. Anaconda hanterar installationen av Notebook vilket skapar en praktisk miljö för arbetet inom Notebook med Python.

3.7 Python

Python är ett kraftfullt och dynamiskt programmeringsspråk anpassat för att vara lättanvänt och tydligt [26]. Python är berikat med ett stort bibliotek av moduler vilket gör det möjligt att användas väldigt fritt. Det innehåller nedladdningsbara biblioteket lämpat till allt från visualisering, datahantering, maskininlärning och mycket mer. Python stöder även bland annat objektorienterad och imperativ programmering vilket gör det anpassningsbart för en stor mängd olika projekt. Kopplat till enkla verktyg för installation av bibliotek kan språket användas för att skapa rörliga projektmiljöer inom allt från forskning till utbildning.

4. Genomförande

4.1 Testning av AutoML-modeller

Innan AutoML-ramverken kunde analyseras och tillämpas krävdes först ett temporärt dataset. Från hemsidan kaggle [1] samlades fem år av historisk OHLC-aktiedata från amerikanska S&P 500 mellan årtalen 2013 fram till 2018. Efter att aktiedata var samlad och redo för användning förbereddes pythonkod för ramverken med syftet att förutsäga nästa dags stängningskurs. Prediktionen grundades på tekniska indikatorer samt tidsberoende variabler baserade på tidigare kursdata. Därför behövdes en uppsättning av variabler konstrueras med ett antal statistiska transformationer och korrekta tekniska indikatorer. Dem viktigaste inkluderar glidande medelvärden (SMA, EMA), momentumindikator (RSI, ROC, MACD) samt volatilitetsmått (ATR, standardavvikelse). Utöver dessa adderades ytterligare ingångsvariabler såsom stapeldiagram och pivotpunkter, dessutom skapades tidsbaserade variabler för exempelvis vecka eller kvartal samt förskjutna variabler.

Denna struktur testades sedan separat på ramverken H2O och Auto-sklearn. Med fokus på prestanda, användarvänlighet och träningstid analyserades och noterades resultaten. För uppdelning av datan användes standardstrukturen för maskininlärning vilket är träningsset och testset. Under dessa initiala tester framstod det tydligt att ett problem hade uppstått i koden: graferna redovisade på alldeles för orealistiska resultat. Problemet berodde på att målvariabeln “close” inte förskjutits korrekt i förhållningen till valda ingångsvariabler vilket ledde till att modellerna förutsåg stängningskursen för samma dag istället för dagen efter. Problemet korreleras dock snabbt vilket ledde till mer realistiska resultat. Hädanefter skiftades fokuset till modellernas resultat. Detta gjordes för att få en bättre förståelse för modellernas tidseffektivitet och skalbarhet, vilket uppnåddes genom att visualisera deras resultat individuellt. Modellerna som utvärderades inkluderade exempelvis trädmodeller, neurala nätverk och ensemble.

4.1.2 Rullande tester

För att analysera olika ramverks tidseffektivitet simulerades ett praktiskt exempel där ny data successivt tillades inom olika intervaller. Ramverken för detta test var H2O, FLAML, Autogluon, Auto-sklearn och Mljar. Målet med detta var att utsätta ramverken för mer krävande tester för att analysera träffsäkerheten i förhållande med tidseffektivitet. För att uppnå detta implementerades ett rullande träningsmönster där ramverkens modeller tränades på den ackumulerade datan från bland annat tidigare veckor för att sedan testas på kommande vecka. Detta innebär att ett test på exempelvis vecka 4 tränades på datan från vecka 1 till 3, intervallerna som valdes inkluderade år, veckor och dagar. Ramverken redovisade alla relativt bra resultat angående precision och träffsäkerhet, de ramverken som var mest tidseffektiva var FLAML samt Auto-sklearn medan H2O och Autogluon redovisade något sämre resultat rent tidsmässigt. Mljar presterade på lägst nivå vilket inte är helt oförväntat med tanke på ramverkets fokus på generering av diverse grafer och liknande tidskrävande funktioner.

4.1.3 Visualisering

För att få en bättre överblick av AutoML-ramverkens resultat skapades under projektets gång en rad av olika visuella hjälpmedel. Dessa inkluderade förvirringsmatriser, stapeldiagram och grafer med och utan datamärkning. Under projektets första fas skapades linjediagram för att redovisa skillnaden mellan det aktuella och förutsedda resultatet av nästa dags stängningskurs. Under senare skede implementerades diverse förvirringsmatriser och stapeldiagram för att visualisera antalet korrekta datamärkningar.

4.2 Datamärkning

För att skapa system som är lämpat för att tolka verkliga marknadsrörelser lades fokus hädanefter på att bygga målvariabler som bättre reflekterade realistiska handelsbeslut med hjälp av trender istället för exakta prisvärden. Detta blev ytterligare påbyggt tillsammans datamärkning för att skapa tydliga signaler för utfallstester med hjälp av att konstruera en handelsstrategi baserat på köp, sälj eller neutrala signaler.

4.2.1 Regressionsuppgifter

Inledningsvis användes uppgifter av typen regression där målvariabeln definierades som nästkommande handeldags stängningskurs. Målvariabeln skapades genom att flytta fram stängningskursen en dag framåt i tiden för varje datapunkt. På så sätt fick varje rad i datasetet en målvariabel som ramverken skulle använda för att göra prediktioner. Variablerna valdes som ett urval av enkla men ändå informativa prisbaserade variabler. Dessa bestod av glidande medelvärden och momentumindikatorer över rullande fönster. Mer specifikt ingick ett tio dagars glidande medelvärde, ett tio dagars exponentiellt medelvärde och ett fjorton dagars relativt styrkeindex. Dessa syftade till att fånga nivå, riktning och hastighet i prisutveckling, vilket var avgörande för att modellerna skulle kunna generera robusta prispredikationer.

4.2.2 Klassifikationsuppgifter

I den senare fasen av projektet användes uppgifter i form av klassifikation istället för regression. Detta användes tillsammans med datamärkning genom att dela upp varje aktieserie i datasetet med fasta fönster om tio handelsdagar, för att sedan ge varje fönster en etikett som köp, sälj eller neutral. För varje fönster anpassades en linjär regressionslinje mot stängningpriserna, där standardavvikelsen av residualerna användes för att sätta en övre samt en undre gräns vid två standardavvikelser. Om fönstrets sista dagspris låg över den övre gränsen fick det etiketten köp, och om priset befann sig under den undre gränsen klassificerades det som sälj. Befann sig priset istället inom kanalen beräknades procentuell förändring mellan fönstrets första och sista dag. Om denna beräkning översteg fyra procent räknades det som köp, understeg den fyra procent räknades det som sälj, annars tillgavs det en neutral etikett alltså varken köp eller sälj.

Utöver datasetets grundliga OHLC-värden samt volym för varje aktie, adderades även daglig pris och volyminformation från fonden SPDR S&P 500 ETF Trust (SPY) till datasetet. Denna data användes i form av regimfilter för att skapa en mer robust och konservativ datamärkning. På den sista dagen i varje segment jämfördes SPYs stängningskurs med dess hundra dagars glidande medelvärde. Ett fönster som tilldelats en köpetikett behöll den endast om SPYs stängningskurs för den sista dagen i fönstret översteg dess glidande medelvärde, och en säljetikett behölls endast om stängningskursen låg under det glidande medelvärdet för SPY. Uppfylldes inte dessa krav sattes etiketten istället till neutral. Detta syftade till att göra modellerna mer vaksamma över marknaden som helhet, eftersom fonden är utformad efter att försöka följa S&P 500-indexet.

Innan segmenteringen av datan adderades även en del tekniska indikatorer till datasetet som relativt styrkeindex med fjorton dagars fönster, glidande medelvärden över trettio respektive femtio dagars fönster och genomsnittligt sant intervall vilket är en indikator som mäter prisvolatiliteten över fjorton dagar. Dessa indikatorer användes i syfte att öka precisionen hos modellerna.

4.3 Utfallstest

I slutfasen av arbetet valdes H2O, FLAML och AutoGluon ut för vidare testning eftersom de visat sig vara de främst presterande ramverken under projektets gång. Utfallstestet användes för att utvärdera modellerna som respektive ramverk tog fram i en verklig miljö genom historisk simulation på tidigare prisdata.

Till testet användes datasetet som beskrivs i 4.2.2, vilket innehöll fjortontusen unika aktier med data över en period på tjugofem år. Från detta fick ramverken träna på ett slumpmässigt urval om tvåtusen aktier och testades därefter på tvåtusen helt nya aktier för att spegla hur väl de fungerar på osedd data. Efter träningen sparades ramverkens respektive bästa modell, testdatan samt klasserna som köp, neutral och sälj. Då ramverken hanterar och sparar dessa på olika sätt behövdes tre separata skript skapas, ett för varje ramverk, för att hämta prediktionerna och omvandla dem till signaler. Varje skript laddade in ramverkets modell samt klasserna tillsammans med testdatan och skapade sedan köp- och säljsignaler. Dessa tre signal-filer användes sedan genom att laddas in till QuantConnect, vilket är en plattform för algoritmisk handel [27], där strategin gick ut på att direkt följa signalerna som modellerna genererat. Därefter kördes testet över en treårig period under samma marknadsförhållanden för att jämföra nyckeltal mellan ramverken.

5. Resultat

5.1 Komplexitet och tolkbarhet

De flesta AutoML-ramverk som testades i projektet har inställningar vilka bestämmer hur lång tid en körning maximalt får ta. Uppnås denna tid innan det har hunnit köra klart, avbryts körningen och den bästa modellen vilken hittats till denna tidpunkt rankas högst. Detta användes frekvent under arbetet då vissa körningar kunde bli väldigt långa, därav baseras denna del främst på när ramverken fått köra med standardinställningar.

Bland de AutoML-ramverk som testades var FLAML helt klart det mest tids- och resurssnåla, vilket ligger i grund för dess begränsade antal basmodeller. Detta innebar att en modell togs fram på kort tid utan att kompromissa allt för mycket med prestandan. Auto-sklearn och Mljar var något långsammare då de byggde något större ensembler samt testade flera basmodeller, men höll sig fortfarande inom en rimlig körningstid. AutoGluon hamnade högre upp på skalan på grund av tiotalet modeller i varje ensemble som byggdes, inklusive flera lager av stacking och bagging av dessa. H2O rankades sämst inom detta då det tar upp väldigt mycket resurser och gång på gång hade längst körningstid. Detta beror på att ramverket tränar en hel mängd olika typer av modeller och flera nivåer av ensembler.

Tolkbarheten kring ramverken skiljer sig åt mycket och i denna kontext syftar det på hur lätt det är att förstå sig på samt hur mycket det förser användaren med, exempelvis en ledartavla eller utskrift. Bäst inom detta var Mljar, där användaren ges en interaktiv ledartavla efter varje körning vilken visar alla modeller som använts och deras respektive prestandamått. Ramverket är framtaget för att vara visualiserande vilket gör det väldigt lättförståeligt. H2O rankas även högt inom detta på grund av en ledartavla som skrivs ut efter varje körning där det är enkelt att se alla modeller och hur de presterat. Det har också en inbyggd funktion där man kan se hur lång tid det är kvar på en körning. FLAML, Auto-sklearn och AutoGluon skriver alla ut mycket information under själva körningen men efteråt visas det inte särskilt mycket. Den mesta informationen sparas i olika filformat och måste hämtas ut manuellt för att få del av vilka modeller som tränats och hur dessa har presterat.

5.2 Användarvänlighet

AutoML-ramverk är skapade för att användas av en bred publik genom att vara användarvänliga. Trots detta skiljer de sig åt i hur mycket förkunskap användaren behöver ha för att kunna navigera ramverken.

Ett av de mest användarvänliga ramverken som testades var FLAML vilket krävde lite manuellt skriven kod för att kunna användas. Användaren anger sitt dataset och sin målvariabel, anropar ramverket och får snabbt en stabil modell utan att behöva förstå det underliggande arbetsflödet. Mljar var också lättförståeligt utan djupare inblick i maskininlärning och dataanalys. Med några få rader kod får man en stabil modell samt skapas en rapport som användaren kan interagera med för att enkelt se vad ramverket har gjort. H2O var också ett av de enklare inom detta där det kombinerar ett Python-API med ett grafiskt gränssnitt. Ramverket går att initiera med endast ett par rader kod, vilket sedan genererar en översiktlig ledartavla över modellerna.

Auto-sklearn är byggt på scikit-learn, vilket gör det enkelt för användare som har förkunskap inom det. Ramverket var dock aningen svårare att använda än tidigare nämnda eftersom det inte är vidare uppdaterat och kräver en äldre variant av Python för att köras. Det går heller inte att köra på ett Windows-system utan måste användas i en Linux-miljö då det är beroende av en specifik Python-modul. AutoGluon var relativt enkelt att använda men resultaten med standardinställningarna var inte på samma höjd som de andra ramverken. För att få liknande resultat krävdes en del optimeringar av ramverket, exempelvis att ändra dess parametrar.

5.3 Prestanda

Under projektets gång har en del AutoML-ramverk testats för dess användning inom finansmarknader. I detta avsnitt presenteras de resultat som uppnåtts under arbetet för både regression- och klassifikationsuppgifter. I tabell 5.1 redovisas rotmedelskvadratfelet samt genomsnittligt absolutfel för regression. Tabell 5.2 avser klassifikation och visar mått som precision, återkallning och f1-poäng. För en mer detaljerad beskrivning av dessa se avsnitt 2.5.

5.3.1 Regression

Utifrån tabell 5.1 presterade H2O klart bäst på det initiala testet medan Mljar låg tätt intill där bakom. FLAML, AutoGluon och Auto-sklearn fick däremot betydligt högre fel. Detta test gjordes tidigt i projektet och användes för att få en känsla kring hur ramverken hanteras och fungerar. Därav användes standardinställningar för respektive av de som testades, vilket visar på att H2O kan hantera tidsserie-specifika mönster i prisdata bäst utan ingående förändringar av data eller ramverkets parametrar.

Ramverk	Rotmedelkvadratsfel	Genomsnittligt absolutfel
FLAML	12.76	1.22
H2O	2.05	0.88
Mljar	5.29	0.98
Auto-sklearn	12.26	1.18
AutoGluon	13.03	1.22

Tabell 5.1: Prediktionsfel för AutoML-ramverk vid förutsägelse av nästa dags stängningskurs på fem års aktiedata

5.3.2 Klassifikation

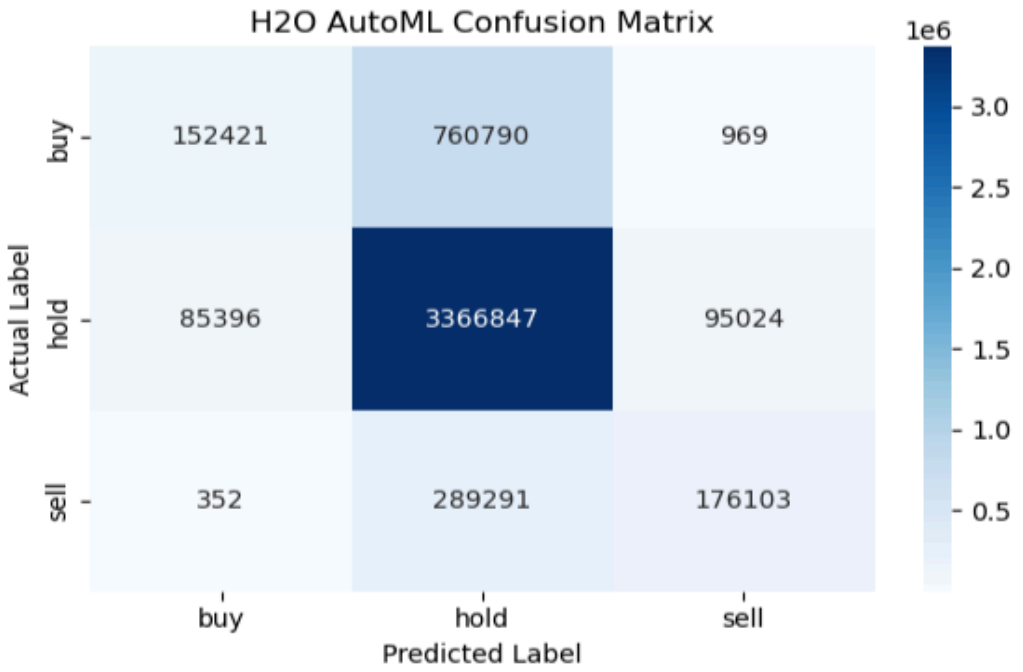
Förvirringsmatriserna, figur 5.1-5.3, visar ett tydligt gemensamt mönster hos alla tre modeller där de identifierar den dominerande neutrala klassen väldigt bra, men sämre på de mer obalanserade köp- och säljklasserna. Detta gjordes avsiktligt med hjälp av regimfilter som omvandlade en del köp- och säljsignaler till neutrala om ett visst villkor inte uppfyldes. Eftersom en neutral position innebär att man avvaktar, och därmed inte riskerar att förlora kapital, är det mer fördelaktigt om osäkerheten kring marknaden eller den specifika aktien är hög. Tabell 5.2 visar de klasspecifika prestandamåtten för respektive ramverk där samtliga har mycket hög återkallning för den neutrala klassen, där H2O har högst med 0.95 följt av 0.90 för FLAML och AutoGluon. Precisionen för den neutrala klassen ligger också på en hög nivå för alla ramverk, vilket visar att det är få falska neutrala prediktioner.

För både köp och sälj klasserna är återkallningen betydligt lägre hos ramverken på grund av regimfiltren. Detta innebär hur många av de faktiska köp- och säljtillfällena som predikteras rätt, där H2O hittade sjutton procent av alla verkliga köp samt trettioåtta procent av alla verkliga sälj. AutoGluon låg snäppet bakom med arton procent vid köp och tjugotre procent för sälj-klassen. FLAML var helt klart bäst på denna aspekt och lyckades fånga upp trettiofem procent för köp samt fyrtiosex procent för sälj. Precisionen för både köp och sälj klasserna var relativt lika för respektive ramverk.

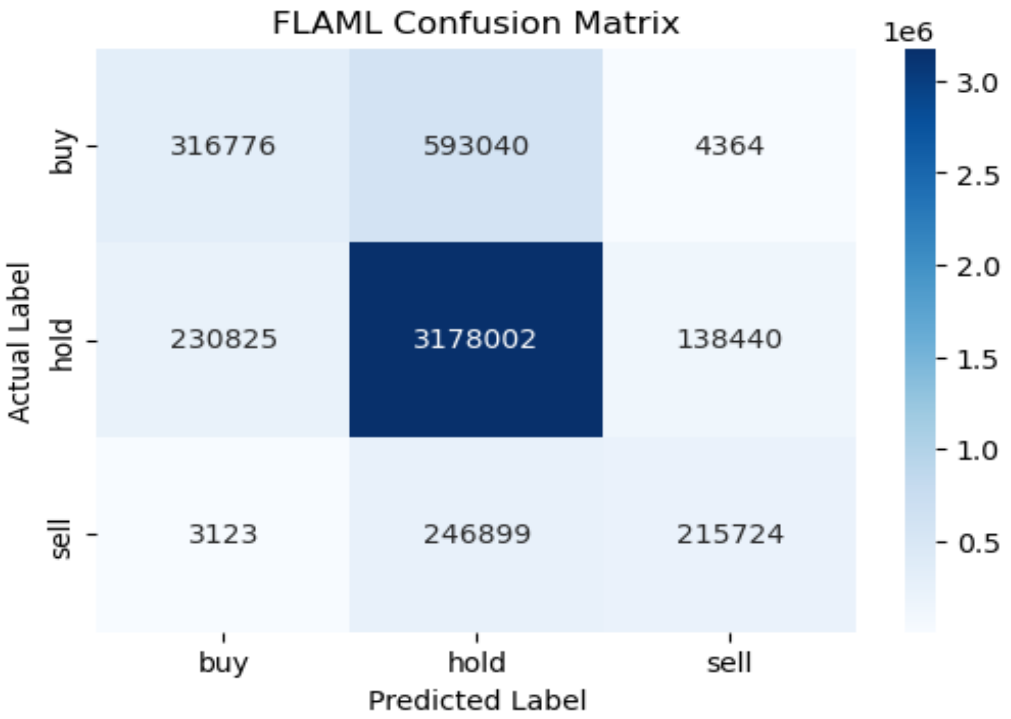
F1-poängen, vilket är en balans av precision och återkallning, blir därav förhållandevis ganska låg för köp- och säljklasserna, men väldigt hög för den neutrala klassen där alla ramverken hade över 0.8. Detta visar på en riskmedveten modell där den hellre predikterar neutral än köp eller sälj vid tillfällen där osäkerheten är hög.

Ramverk	Klass	Precision	Återkallning	F1-poäng
H2O	Köp	0.64	0.17	0.26
H2O	Neutral	0.76	0.95	0.85
H2O	Sälj	0.65	0.38	0.48
FLAML	Köp	0.58	0.35	0.43
FLAML	Neutral	0.79	0.90	0.84
FLAML	Sälj	0.60	0.46	0.52
AutoGluon	Köp	0.61	0.18	0.28
AutoGluon	Neutral	0.73	0.95	0.82
AutoGluon	Sälj	0.63	0.23	0.34

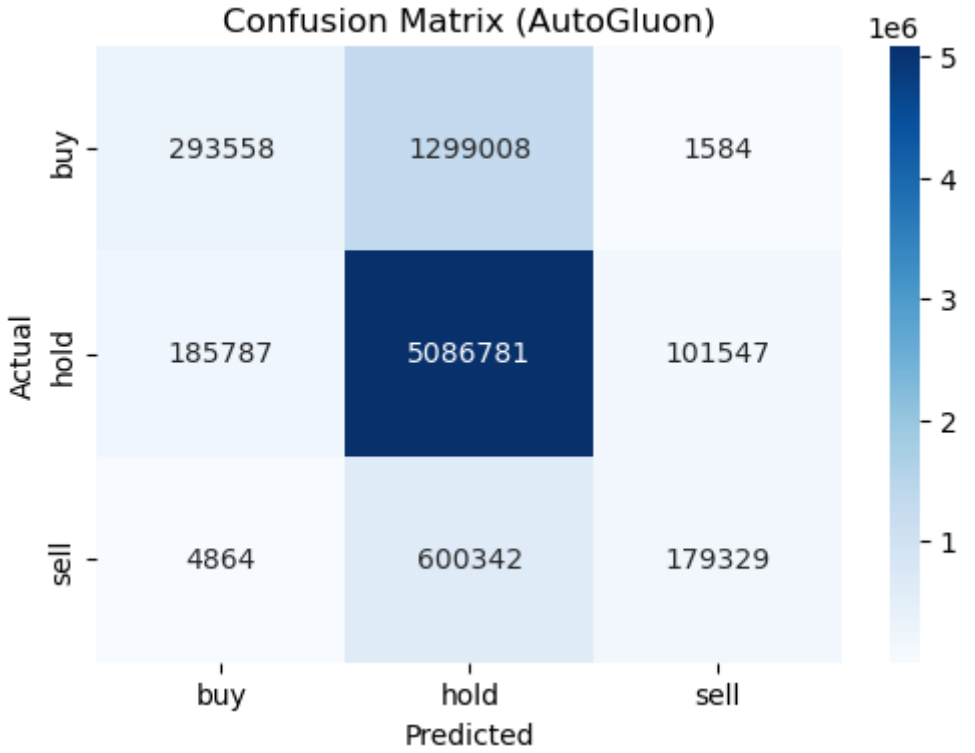
Tabell 5.2: Klasspecifika prestandamått för H2O, FLAML och AutoGluon vid mångfaldig klassifikation



Figur 5.1: Förvirringsmatris för H2O vid klassificering av köp, neutral och sälj



Figur 5.2: Förvirringsmatris för FLAML vid klassificering av köp, neutral och sälj



Figur 5.3: Förvirringsmatris för AutoGluon vid klassificering av köp, neutral och sälj

5.3.3 Utfallstest

Från figurer 5.4-5.6 framgår det att H2O uppnådde högst avkastning med 326 % och en vinstprocent på 48 %. FLAML nådde en något lägre avkastning på 243 % men en högre vinstprocent på 55 % vilket visar på en mer jämn och stabil vinst per affär. AutoGluon presterade helt klart sämst av de tre ramverken med endast en avkastning på 47 % och en vinstprocent på ungefär 13 %. Kapitalutvecklingen, vilket är själva kurvan i figurerna, visar att H2Os modell drar ifrån under de senare åren och med relativt små nedgångar. FLAMLs modell uppvisar istället en ganska jämn uppåtgående kurva med få nedgångar under testet vilket indikerar att den hanterar marknaden väl och genererar en avkastning med mindre risk. AutoGluons kurva är däremot väldigt fluktuerande och når aldrig samma trendmässiga uppsving som de andra två, vilket visar på att den har svårt att urskilja stabila signaler utifrån prisdatan.

Sammanfattningsvis bekräftar detta att ramverkens förmåga att kunna identifiera neutrala signaler på en hög nivå är avgörande för att generera en stabil avkastning med mindre risk samt undvika förluster vid osäkra tillfällen.

Figur 5.4: Historisk simulering med H2O över en period på tre år

Figur 5.5: Historisk simulering med FLAML över en period på tre år

Figur 5.6: Historisk simulering med AutoGluon över en period på tre år

6. Diskussion

6.1 Tolkning av komplexitet och tolkbarhet

Komplexiteten kring AutoML-ramverken som använts under projektet varierar ordentligt. Resultaten visar att FLAML utmärker sig som det mest resurssnåla alternativet genom dess begränsade antal basmodeller, vilket kan leverera en robust modell på kort tid. Detta gör att ramverket är särskilt lämpat för användning vid analyser där tidsaspekten är viktigare än att få fram den mest optimala prestandan. I mittenskiktet hamnade Mljar, Auto-sklearn och AutoGluon. Dessa tre bygger alla någorlunda stora ensembler och tester fler basmodeller än FLAML, men håller sig fortfarande inom en rimlig tidsram vid varje körning. H2O rankas lägst på denna aspekt då det tränar många olika typer av modeller samt flera nivåer av ensembler. I testerna från resultatet gav detta längst körningstider men även bra prestanda. Därav är det lämpat för projekt där prestanda väger tyngre än effektivitet.

Utifrån resultaten var Mljar överlägset bäst på att tolkas tack vare dess interaktiva ledartavla som presenteras efter varje körning. Den visar översiktligt alla testade modeller med tillhörande nyckeltal inom prestanda, vilket gör det enkelt att snabbt förstå vilka typer av modeller som använts och hur de presterat utan förkunskap kring maskininlärning. Även H2O kunde tolkas relativt enkelt på grund av sin ledartavla som också skrivs ut efter varje körning. Där får användaren direkt en tabell över respektive modell och hur de presterat. Till skillnad från dessa två ramverk levererar FLAML, Auto-sklearn och AutoGluon bra utskrift under själva körningen om vad som händer men sparar sedan sammanställningen i olika filformat utan automatisk presentation. Om användaren vill tolka resultaten ytterligare behöver man läsa in detta manuellt och bygga egna visualiseringar. Vi anser därför att dessa tre ramverk är mer lämpade för användare med tidigare förkunskap inom ämnet.

6.2 Reflektion kring användarvänlighet

Arbetet med de olika AutoML-ramverken under projektet visade att FLAML och Mljar var de lättaste att komma igång och arbeta med. Dessa passar bra i lärande eller explorativa sammanhang där användaren snabbt vill lära sig hur dessa ramverk fungerar på en ytlig nivå. H2O var även ett av de lättare att förstå sig på. Med dess gränssnitt kan både nybörjare och dataanalytiker enkelt hantera det på ett stabilt plan.

AutoGluon var lite besvärligare att använda samtidigt som standardinställningarna gav relativt dåliga resultat. Därav är det ett ramverk som bör användas av de som har förkunskap inom området för att nyttjas på det mest optimala sättet. Auto-sklearn var det mest problematiska att använda i och med dess förankring till gamla Python-versioner. Det går heller inte att använda på Windows-system vilket gör det mindre lämpat i allmänhet. På grund av detta samt mediokra resultat som ramverket gav under projektet rekommenderar vi det inte för vidare användning.

6.3 Diskussion om prestanda

De tidiga testerna i projektet där ramverken skulle förutspå nästa dags stängningskurs för olika aktier visar att H2O hanterar dessa typer av problem bäst. Det lyckades även uppnå bäst resultat på de rullande testerna som vi gjorde därefter där nyare data adderades under körningarna. Mljar presterade även relativt väl på dessa tester och låg ganska nära H2O resultatmässigt. Skillnaderna mot FLAML, AutoGluon och Auto-sklearn var tydliga där dessa ramverk levererade högre fel under samma villkor. Målet med detta var att få en initial känsla av hur AutoML-ramverk fungerar och hanterar rå prisdata utan vidare förbehandling, där resultatet indikerar att H2O bör användas för dessa typer av uppgifter.

Därefter utfördes tester av typen klassifikation där H2O, FLAML och AutoGluon valdes för vidare utvärdering då vi ansåg dessa vara bäst utifrån ett antal kriterier. Där visade alla ramverk att de kunde identifiera den neutrala klassen på en hög nivå. H2O och AutoGluon hade det dock betydligt svårare att fånga upp de verkliga köp- och säljtillfällena. FLAML stack där ut genom att fånga upp många av dessa trots att dess totala felmarginal var högre än H2Os. Precisionen var ganska hög och likartad för respektive ramverk vilket visar på att när de väljer att genomföra en transaktion görs det vid rätt tillfälle. Dessa resultat visar på skillnader i hur ramverken balanserar noggrannhet mot träffar kring köp- och säljsignaler. H2O prioriterar att minimera fel över alla klasser medan FLAML, trots högre genomsnittsfel, är bättre på att fånga upp signaler från de mer obalanserade klasserna. AutoGluon hamnade åt samma håll som H2O och lyckades heller inte fånga signalerna på ett tillräckligt bra sätt.

Den historiska simuleringen på tidigare prisdata visade på hur modellerna hade presterat i en verklig miljö. H2O levererade där högst avkastning samt en vinstprocent på 48 %, vilket visar på att den kan hitta långsiktigt vinstgivande mönster utan större nedgångar. Detta ramverk är därav användbart när högst avkastning eller mest prestanda söks, och tidsaspekten inte väger lika tungt. FLAML gav en mer jämn och stabil modell med en avkastning på 243 % samt den högsta vinstprocenten på 55 %. Den höga vinstprocenten indikerar att modellen träffar rätt fler gånger vid faktiska handels tillfällen. Den jämna kurvan med få kraftiga nedgångar indikerar att FLAML kan vara ett smart val av ramverk där låg risk och stabil avkastning värderas högt. AutoGluon gav sämst resultat med mycket lägre avkastning och vinstprocent än de andra två. Den mycket ojämna kapitalkurvan talar för vår tidigare observation om att ramverket har svårt att urskilja signaler i prisdata utan omfattande ändringar av dess parametrar.

6.4 Resultatets bidrag till företaget

Resultatet kommer vara till nytta som underlag hos företaget genom rekommendationer om vilka AutoML-ramverk som bör användas, inte endast för finansiell dataanalys utan även på ett generellt plan för olika typer av uppgifter. Genom utvärdering av ett antal aspekter hos ramverken visade resultaten att H2O bör användas för de flesta typer av uppgifter medan FLAML är ett bra alternativ om tidsaspekten är den viktigaste för användaren. Mljar kan även övervägas som ett alternativ om användaren inte besitter tidigare kunskap inom maskininlärning eller AutoML och vill lära sig mer på ett intuitivt sätt. Utöver detta kommer projektets arbete ligga till grund för framtida studenter, där dessa ska fortsätta på samma spår och använda materialet för att djupdyka ytterligare inom området.

6.5 Etik

Även om AutoML-ramverken för sig är effektiva finns det alltid en risk att användaren upplever förvirring eftersom grunderna till besluten kan vara svårtolkade för icke-experter. Detta kan leda till brist på förtroende för systemet och resultaten. En av fördelarna med AutoML är att det är enkelt att använda även för icke-experter, men ur ett etiskt perspektiv kan detta vara en nackdel. Dessa utvecklare riskerar att skapa otestade system som inte följer branschspecifika krav. Inom den finansiella sektorn kan detta leda till kapitalförluster på grund av bias.

En annan viktig punkt är dataskydd och integriteten kring användning av AutoML. Ramverken hanterar ofta stora datamängder, vilket kan innehålla kunddata med känslig information om företag eller privatpersoner. Då ramverken testar många olika modeller och parametrar automatiskt kan data exponeras i sparandet av filer eller interna kopior av dessa, vilket ökar risken för dataläckor. Detta gör det viktigt med strikt kontroll över hur data lagras, vem som har tillgång till den samt hur länge det sparas. AutoML kräver också stora mängder datorkraft som i sin tur leder till högre energiförbrukning och mer koldioxidutsläpp. För just detta projekt har träningen av modeller skett på en mindre skala vilket inte är lika påfrestande för miljön. Men för större projekt som kräver längre träning och fler tester kan miljöpåverkan vara ohållbar.

7. Slutsats

I detta projekt har en rad AutoML-ramverk utvärderats och jämförts med avseende på komplexitet, tolkbarhet, användarvänlighet och prestanda. Resultaten visar att H2O är ett mångsidigt ramverk med bra prestanda på alla typer av uppgifter, medan FLAML med dess lätta konfiguration är anpassbar för att snabbt ta fram modeller som håller en hög nivå. Mljar sticker ut med sin interaktiva ledartavla och är därav ett smart val för användare utan tidigare kunskap inom dataanalys. AutoGluon och Auto-sklearn kräver betydligt mer underhållning för att nå samma nivåer som tidigare nämnda ramverk, därav rekommenderas de inte för vidare användning.

Utifrån detta rekommenderas både H2O och FLAML som två stabila ramverk för ytterligare testning samt användning. Vill man snabbt ha en modell som ger hög prestanda utan att förstå det underliggande arbetsflödet bör FLAML väljas. Om inte tidsaspekten är den viktigaste och högst prestanda söks är H2O ett bra val. Mljar rekommenderas också som ett yterliggare val ifall ändamålet är lärande med dess interaktiva och visualiserande rapporter.

8.Källförteckning

[1] A. Goldbloom och B. Hamner, Kaggle.com.
Tillgänglig: https://www.kaggle.com/
[2] T. O. Hodson, “Root-mean-square-error (RMSE) or mean absolute error (MAE): when to use them or not”, Geoscientific Model Development, vol. 15. nr. 14, ss. 5481-5487. Jul. 2022. Tillgänglig: https://gmd.copernicus.org/articles/15/5481/2022/
[3] V. Jayaswal, “Performance Metrics: Confusion matrix, Precision, Recall and F1 Score”, Towards Data Science, Sep. 14, 2020. Tillgänglig: https://towardsdatascience.com/performance-metrics-confusion-matrix-precision-recall-and-f1-score-a8fe076a2262/
[4] F. Hutter, L. Kotthoff och J. Vanschoren, “Automated Machine Learning", Springer International Publishing, 2019. Tillgänglig: https://link.springer.com/book/10.1007/978-3-030-05318-5
[5] N. Buslim, I. L. Rahmatullah, B. A. Setyawan och A. Alamsyah, “Comparing Bitcoin’s Prediction Model using GRU, RNN and LSTM by Hyperparameter Optimization Grid Search and Random Search”, 9:e International Conference on Cyber and IT Service Management, Sep. 2021, ss. 1-6. Tillgänglig: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9588947
[6] D. E. Puentes G., C. J. Barrios H. och P. O. A Navaux, “Hyperparameter Optimization for Convolutional Neural Networks with Genetic Algorithms and Bayesian Optimization”, Latin American Conference on Computational Intelligence, 2022. ss. 1-5. Tillgänglig: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9981104
[7] J. Groß, Linear Regression. Springer Nature, 2003. Tillgänglig: https://link.springer.com/book/10.1007/978-3-642-55864-1
[8] D. G. Kleinbaum och M. Klein, Logistic Regression. Springer New York, 2010. Tillgänglig: https://link-springer-com.proxy.lib.chalmers.se/book/10.1007/978-1-4419-1742-3
[9] J. H Friedman, “Greedy function algorithm: A gradient boosting machine”, The Annals of Statistics, vol. 29, nr. 5, ss. 1189-1232. Okt. 2001. Tillgänglig: https://doi.org/10.1214/aos/1013203451
[10] T. Chen och C. Guestrin, “XGBoost: a Scalable Tree Boosting System” Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD ’16, vol. 1, nr. 1, ss. 785–794, Aug. 2016. Tillgänglig: https://dl.acm.org/doi/pdf/10.1145/2939672.2939785
[11] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, “Lightgbm: A highly efficient gradient boosting decision tree”, Advances in neural information processing systems, ss.3146-3154, 2017. Tillgänglig: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf
[12] A. Dorogush, V. Ershov och A. Gulin, “Catboost: gradient boosting with categorical features support", Yandex, 2018. Tillgänglig: https://arxiv.org/pdf/1810.11363
[13] V. K. M. Nagaraju, V. A. G. Nicholas, S. Raju och T. Jayaraman, “A Robust Breast Cancer Classification System Using Multilayer Perceptron and Grey Wolf Optimization”, Traitement du Signal, vol. 42, Feb. 2025.

https://web.p.ebscohost.com/ehost/detail/detail?vid=0&sid=79bb5765-6faf-4b1d-b4b0-29642a523e23%40redis&bdata=JkF1dGhUeXBlPXNzbyZhdXRodHlwZT1zc28mY3VzdGlkPXMzOTExOTc5JnNpdGU9ZWhvc3QtbGl2ZSZzY29wZT1zaXRl#db=bsu&AN=183569741
[14] Y. G. Jahed, S. Y. M. Mousavi och S. Golestan, "Online Stream-Driven Energy Management in Microgrids Using Recurrent Neural Networks and SustainaBoost Augmentation", IEEE Transactions on Sustainable Energy, vol. 16. nr. 2, ss. 1153-1164. Apr. 2025. Tillgänglig: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10768873
[15] E. Budu, "Bagging, Boosting and Stacking in Machine Learning", Baeldung, Maj. 2025. Tillgänglig: https://www.baeldung.com/cs/bagging-boosting-stacking-ml-ensemble-models
[16] "Introduction to Algorithms - A comprehensive Guide for Beginners - Unlocking Computational Thinking", LLC Cuantum Technologies, 2023. Tillgänglig: https://app.knovel.com/kn/resources/kpIAACGBU1/toc
[17] R. Li, "What is Big O Notation" Medium, Dec. 30, 2019. Tillgänglig: https://medium.com/@richardli125/what-is-big-o-notation-d72f602370ea
[18] T. Dokeroglu, A. Deniz och H. E. Kiziloz, "A comprehensive survey on recent metaheuristics for feature selection", Neurocomputing, vol. 494, ss. 269-296. Tillgänglig: https://www.sciencedirect.com/science/article/pii/S092523122200474X
[19] F. Trotta, "Understanding l1 and l2 Regularization" Towards Data Science, Maj. 2022. Tillgänglig: https://towardsdatascience.com/understanding-l1-and-l2-regularization-93918a5ac8d0/
[20] C. Wang, Q. Wu, M. Weimer och E. Zhu, "FLAML: A Fast and Lightweight AutoML Library", arXiv.org, 2019. Tillgänglig: https://arxiv.org/pdf/1911.04706
[21] E. Ledell och S. Poirier, "H2O AutoML: Scalable Automatic Machine Learning", 7th ICML Workshop on Automated Machine Learning, 2020. Tillgänglig: https://www.automl.org/wp-content/uploads/2020/07/AutoML_2020_paper_61.pdf
[22] "AutoML mljar-supervised", Mljar.com, 2024. Tillgänglig: https://supervised.mljar.com/
[23] M. Feurer, A. Klein, K. Jost, T. Springenberg, M. Blum och F. Hutter, "Efficient and Robust Automated Machine Learning", Tillgänglig: https://papers.nips.cc/paper_files/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf
[24] N. Erickson, J Mueller, A. Shirkov, H. Zhang, P. Larroy, M. LI och A. Smola, "AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data", Amazon Web Services, Mar. 2020. Tillgänglig: https://arxiv.org/pdf/2003.06505
[25] Anaconda Software Distribution, version 2-2.4.0. Anaconda, nov. 2016. Tillgänglig: https://www.anaconda.com/
[26] Python, "About Python", python.org. Tillgänglig: https://www.python.org/about/
[27] QuantConnect, "QuantConnect - Algorithmic Trading Platform", quantconnect.com. Tillgänglig: https://www.quantconnect.com/