CHALMERS

Generering och visualisering av tidsserieprediktioner.

Examensarbete inom Data- och Informationsteknik

Anton Wester
Oscar Nilsson

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2019

Generering och visualisering av tidsserieprediktioner.

Anton Wester
Oscar Nilsson

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET

Göteborg 2019

Generering och visualisering av tidsserieprediktioner.

Anton Wester
Oscar Nilsson

Examinator: Jonas Duregård

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet
412 96 Göteborg
Telefon: 031-772 1000

Institutionen för Data- och Informationsteknik
Göteborg 2019

Sammanfattning

Ett IoT-företag som tillverkar och säljer produkter för hemautomation sparar mycket data från deras installerade produkter som inte nyttjas i företagsrelaterade beslut. Målet med projektet är att undersöka om det finns ett utvecklat API eller en metod för att generera prediktioner baserade på företagets historiska data för att kunna ta mer realistiska företagsbeslut. De tekniska delarna av arbetet har utförts i företagets lokaler, där det har funnits tillgång till datorer och konsultation av erfarna utvecklare, medans de “icke-tekniska” delarna har utförts på Chalmers.

Ett lämpligt API för generering av tidsserieprediktioner lyckades hittas efter veckor av undersökning. API:et är skapat av BigML och tillhandahåller funktionalitet för att läsa in data, bearbeta datan och producera prediktioner. Efter att gruppen bestämt sig för att använda BigML så implementerade de in dess funktionalitet i företagets befintliga analysverktyg och visualiserade den historiska datan tillsammans med prediktionen i en graf.

Gruppen stötte på en del motgångar under projektets gång, men är överlag nöjd med resultatet. En tydlig representation av den historiska datan och den prognostiserade datan lyckades implementeras i företagets kodbas, men det råder osäkerhet om tillförlitligheten på de prognostiserade värdena.

Nyckelord: Tidsserieprediktion, BigML

Abstract

This project was established together with an IoT-company that manufactures and sells products for home automation and saves a lot of data from their installed products which is not used in company-related decisions. The aim of the project is to investigate whether there is a developed API or a method for creating forecasts on the company's historical data in order to be able to make more realistic business decisions. The technical parts of the work have been carried out in the company's premises, where there has been access to computers and consultation of experienced developers, while the "non-technical" parts have been performed at Chalmers.

An API for the creation of time series forecasts was discovered after weeks of investigation. The API is created by BigML and provides functionality to load data, process the data and produce forecasts. After the group decided to use BigML, they implemented its functionality in to the company's existing analysis tool and visualized the historical data along with the forecasted data in a graph.

The group encountered a number of problems during the project, but are generally satisfied with the result. A clear representation of the historical data and the forecast data was successfully implemented in the company's code base, but there is uncertainty about the reliability of the forecasted values.

Keywords: Time-series forecasting, BigML

Förord

Denna rapport är vårt examensarbete på högskoleingenjörsprogrammet inom Datateknik vid Chalmers Tekniska Högskola. Arbetet har genomförts av Anton Wester och Oscar Nilsson under vårterminen 2019. Examensarbetet omfattar 15 högskolepoäng.

Tack till

Under arbetets gång har vi haft stöd av vår handledare Koen Claessen som vi vill tacka för givande råd, värdefull feedback och berikande konversationer. Vi vill även passa på att tacka de involverade hos det företag som vi gjort vårt examensarbete i samarbete med men som vi av rättsliga skäl inte kan nämna vid namn i rapporten. Slutligen vill vi tacka opponenten Erik Tran som gav oss konstruktiv kritik.

Arbetet har inneburit omfattande undersökning kring tidsserieanalyser samt utveckling av funktionalitet kring tidsseriprediktioner i ett analysverktyg. Examensarbetet tillkom genom dialog med ett IoT företag i Göteborgsregionen.

Innehållsförteckning

1. Inledning

Idag besitter många företag stora mängder data men beslut kring produktion och dylikt tas ofta utan datan i beaktning. Detta resulterar många gånger i att beslut kring t.ex. produktion inte överensstämmer med efterfrågan. Detta projekt ämnar att underlätta vid företagsbeslut genom att generera och visualisera tidsserieprediktioner.

1.1 Bakgrund

Det företag som projektet har utförts i samarbete med är i grund och botten ett renodlat IoT-företag som är beläget i Göteborg. De handahåller utveckling, tillverkning och försäljning av produkter för hemautomation som kan nyttjas i bostäder likväl som kommersiella sammanhang. Företaget kommer inte nämnas vid namn i denna rapport då de har bett om att få vara en anonym samarbetspartner för projektet.

Vid installation och användning av deras produkter så samlar företaget in mängder av data i Elasticsearch databaser[1][2]. Vid den inledande fasen av projektet så nyttjade de inte datan från dessa databaser vid produktionsbeslut då de inte hade implementerat någon metod för att bilda prediktioner baserade på den data de har tillgång till. Detta resulterar i att deras produktionsbeslut ofta togs utan ordentligt underlag vilket kan resultera i att estimeringen av produktionen skiljer sig markant från den faktiska efterfrågan.

Projektet ämnade således att underlätta vid produktionsbeslut genom att implementera en vy för tidsserieprediktioner i ett av deras befintliga analysverktyg med hjälp av tidsserieanalyser.

1.2 Syfte

Det primära syftet med projektet är att effektivisera företagets nyttjande av data insamlad kring deras produktlinje för att underlätta vid produktionsrelaterade beslutstaganden. Detta mål ska nås genom att implementera en vy för prediktioner i ett befintligt analysverktyg. För att det primära syftet ska kunna realiseras så kommer en undersökning och utvärdering av befintliga metoder, verktyg och API:er vara fundamentalt för att finna en lösning som passar företagets behov.

1.3 Frågeställningar

Detta är de frågeställningar som togs fram i början av projektet och som kommer att besvaras i denna rapport.

- Vilka metoder är lämpliga för tidsserieprediktioner baserat på företagets data?
- Vilken data hos företaget är relevant att göra prognoser på?
- Är det möjligt att skapa pålitliga prognoser med företagets data?

1.4 Avgränsningar

Dessa avgränsningar togs fram under projektets gång för att tydliggöra vad projektet inte skulle innefatta.

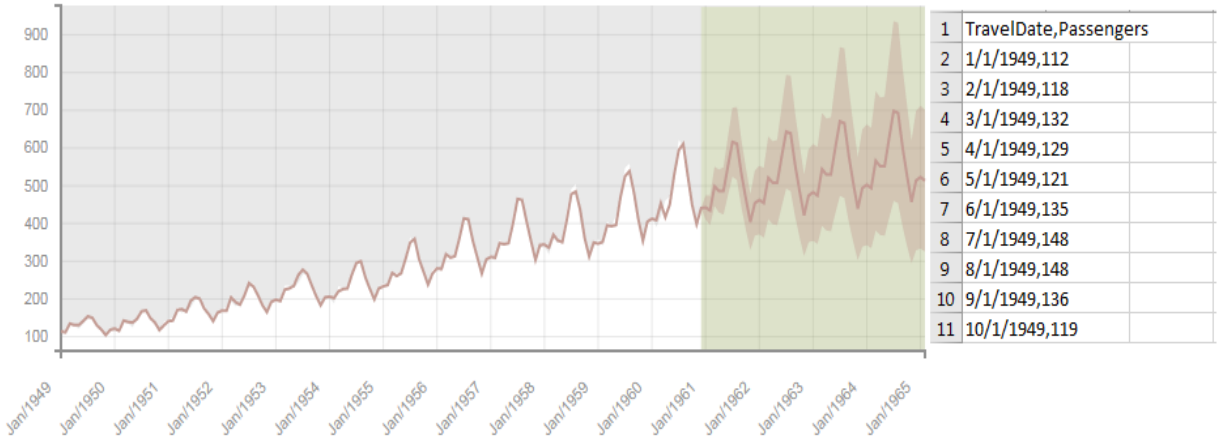
- För projektet kommer inte egen data samlas in utan projektet ämnar enbart att nyttja företagets insamlade data samt testa på data som är ämnad för denna typ av uppgift.

- Verktyg, bibliotek, utvecklingsspråk eller design kring projektet kommer inte kunna väljas fritt då det som utvecklas är tvunget att nyttja företagets nuvarande kodbas till högsta grad och följa den design som de har i sitt analysverktyg.
- Ingen egen databas kommer sättas upp utan istället kommer företagets existerande ElasticSearch databas användas.
- All data ur företagets databas kommer inte nyttjas för att bilda prediktioner utan enbart den data som har att göra med antal dagliga installationer av deras enheter. Detta val gjordes på grund av att det var den data som företaget ansåg vara mest relevant för deras bestlutstaganden.

2. Teknisk Bakgrund

2.1 Tidsserieanalys

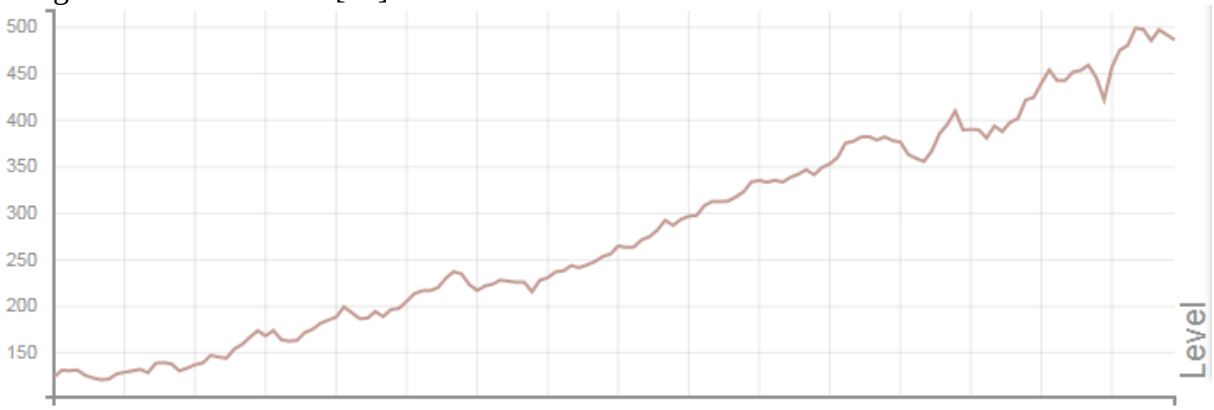
En tidsserie är en samling av datapunkter som har skapats genom att observera en variabels värde under tid med ett regelbundet intervall. Tidsserieanalys utgår från att analysera tidsserier för att utvinna karakteristiska drag i datasetet. Med hjälp av dessa karakteristiska drag kan tidsserieprediktioner bildas för att med hjälp av matematiska modeller baserade på extrapolering försöka förutspå framtida värden baserade på den historiska datan. [3][4] Extrapolering är att estimera värdena utanför ett befintligt mätområde, t.ex. att dra slutsatser om framtida värden baserade på historiska värden.[5]



Figur 1. Exempel på en tidsserieprediktion som är skapad baserad på datan till höger i bild som innehåller datum och hur många passagerare som månatligen har rest med flyg mellan år 1949-1961

2.1.1 Tidsseriens komponenter

Tidsserier kännetecknas av dessa fyra komponenter
Nivå **-** Det lokala genomsnittliga värdet för en tidsserie.
Nivån beräknas t.ex. med hjälp av exponentiell utjämning. Som namnet antyder så minskar exponentiell utjämning variationen i en tidsserie vilket resulterar i en jämnare kurva. Vid exponentiell utjämning så är det filtrerade värdet en viktad summa av alla de föregående värdena där de senaste värdena har den högsta vikten och där vikten exponentiellt minskar när den går mot äldre värden. [40]



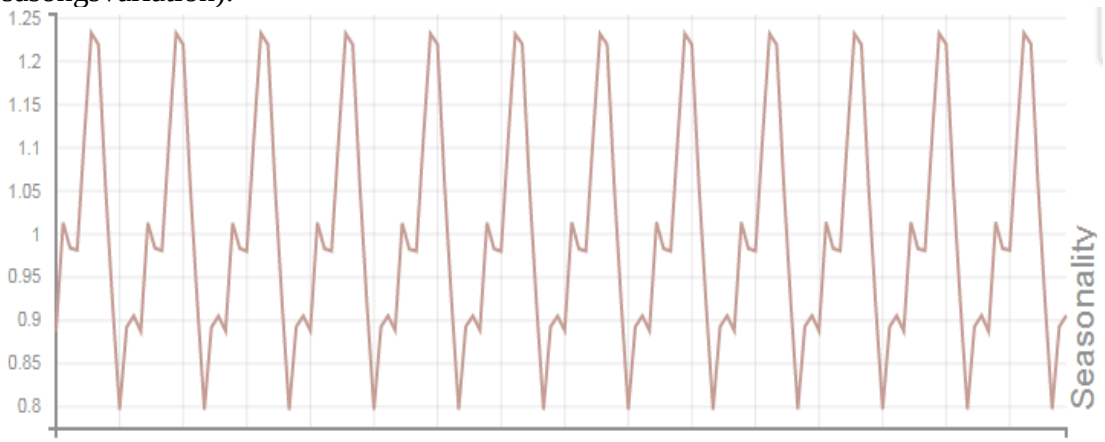
Figur 2. Exempel på nivå genererad av BigML baserad på tidsserien i Figur 1.

Trend **-** Det ökande eller minskande värdet i en tidsserie.
Medan nivån representerar det lokala genomsnittliga värdet av en tidsserie så representerar trenden den långsiktiga förändringen av värdet för en tidsserie. Trenden representeras antingen som differensen mellan på varandra följande nivåvärden(additiv trend, linjär bana) eller som förhållandet mellan dem(multiplikativ trend, exponentiell bana).

Figur 3. Exempel på additiv trend genererad av BigML baserad på tidsserien i Figur 1.

Säsongsvariation **-** Den återkommande kortsiktiga cykeln i en tidsserie.
Säsongsvariation för en tidsserie representerar all variation där värdets variation följer ett konsekvent mönster över tid med intervaller av en specifik längd.
T.ex. så kan försäljning av glass vara högre under sommaren och lägre under vintern år efter år.

Denna variation kan bli modellerad som en relativt konstant mängd oberoende av tidsseriens nivå (additiv säsongsvariation), eller som en relativt konstant andel av nivån (multiplikativ säsongsvariation).



Figur 4. Exempel på multiplikativ säsongsvariation genererad av BigML baserad på tidsserien i Figur 1.

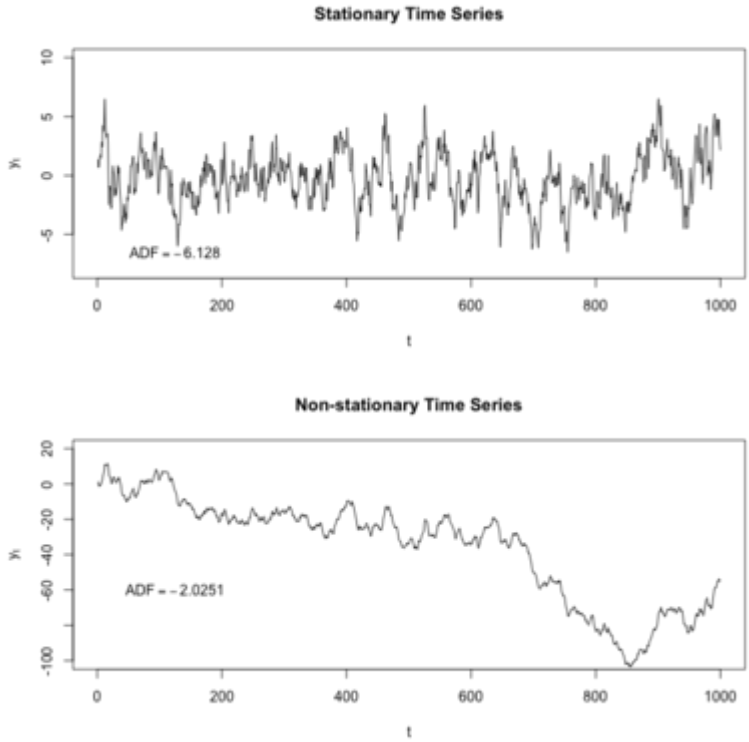
Slumpmässig variation **-** Slumpmässig variation i tidsserien.
Efter att ha tagit nivån, trenden och säsongsvariation i beaktning så kan viss variation som ej är redovisad i modellen existera. Som säsongsvariation så kan slumpmässig variation modelleras som en additiv process eller som en multiplikativ process. Parametrisering av den slumpmässiga variationen är viktig för att beräkna konfidensgränser för tidsserier. [6]

2.1.2 Stationär data

För att data ska kunna nyttjas för att kunna generera någorlunda pålitliga tidsserieaprediktioner så krävs det för de flesta metoder att datan i tidsserien är stationär.[7] För att en tidsserie ska vara stationär så medför det att observationerna i tidsserien inte är tidsberoende. Vilket innebär att tidsserien är stationär om den inte har några trender eller säsongsvariationer och att sammanfattande statistik som t.ex. medelvärdet och variationen för tidsserien är konsekvent över tid.

För att undersöka om en tidsserie är stationär så finns det tre huvudsakliga metoder.

- ***Visuellt Granska Grafer.***
  Manuell granskning av en visuell representation av dataset kan utföras för att finna uppenbara trender och säsongsvariationer.
- ***Sammanfattande Statistik.***
  Undersökning av sammanfattande statistik för datan eller slumpmässigt valda partitioner för att leta efter uppenbara eller signifikanta skillnader. Detta är något som både kan utföras manuellt samt med hjälp av olika verktyg. [41]
- ***Statistiska Test.***
  Att nyttja statistiska test för att undersöka om förväntningarna av stationäritet är uppfyllda eller brutna. Exempel på denna typen av test är Augmented Dickey-Fuller test.[8]



Figur 5. Exempel på stationär och icke stationär tidsserie.[42]

I Figur 5 visas två tidsserier varav den ena är stationär och den andra är icke-stationär. Augmented Dickey Fuller metoden rapporterar för den nedre tidsseriens teststatistik ett resultat som medför att icke-stationäritet inte kan uteslutas.[43]

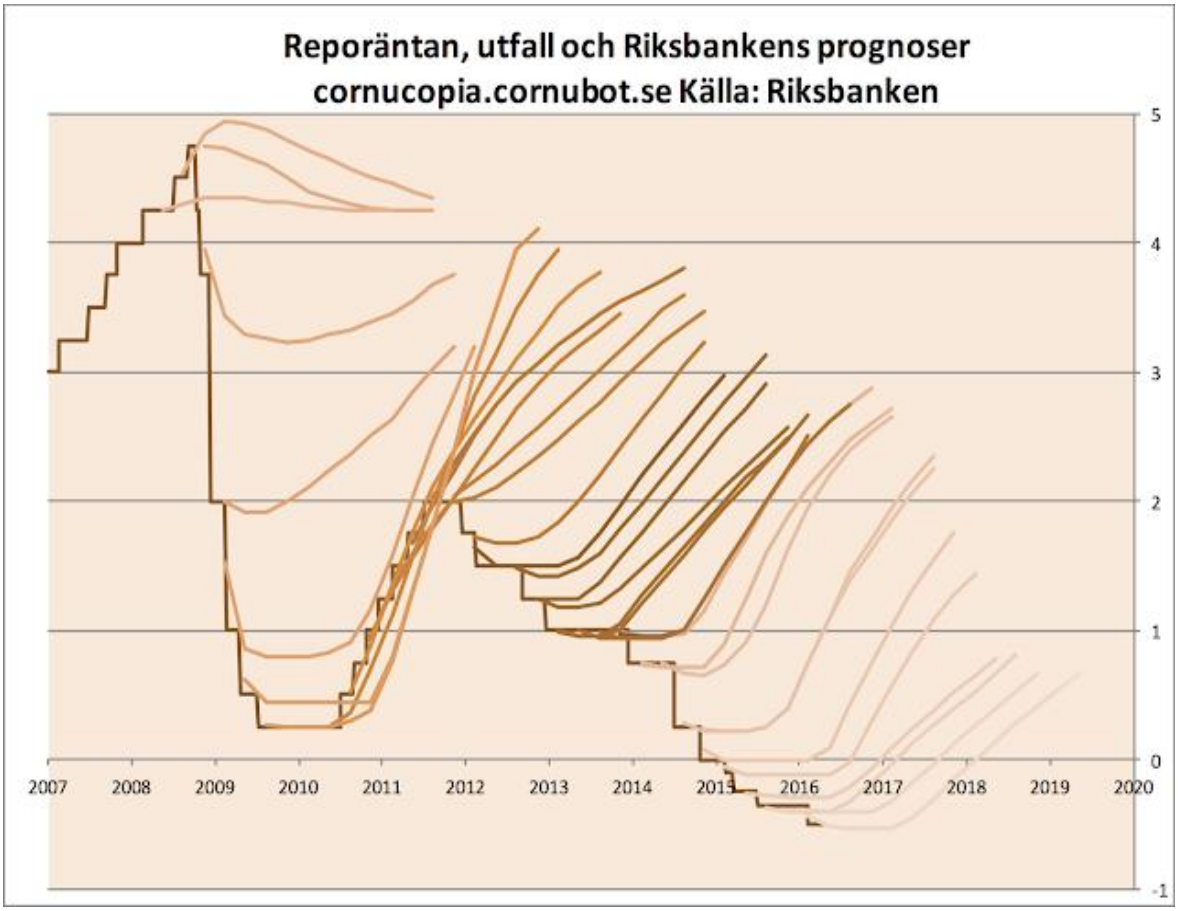
2.1.3 Behandling av data

Att omvandla ett icke stationärt dataset till stationärt kan kräva en hel del manipulering av datan. Några av de vanligaste metoderna som tillämpas vid försök att stationarisera ett dataset är följande:

- Differentiering - Beroende på ordning (n) av en icke-stationär tidsserie kan den differentieras (n) gånger vilket kan resultera i en stationär serie.
- Variansstabilisering - Om variansen inte bedöms vara konstant. Transformeras på samma sätt som vid regressionsanalys, oftast med logaritmering.

Dessa steg var inget som projketgruppen behövde ta i beaktning vid nyttjande av data för projektet då BigML automatiskt bearbetar data vid generering av tidsserieprediktioner.

2.1.4 Säkerhetsnivå



Figur 6. Visualisering av riksbankens räntebana(mörkbruna linjen med start år 2007) samt utav de olika prediktioner som riksbankens tidsserieexporter har utfört vid olika tidpunkter.[39]

Prediktioner kring tidsserieanalayser är ett område som präglas av mycket osäkerhet kring resultat och det är sällan som prediktioner överensstämmer väl med verkligheten över tid. Ett exempel på detta är de prediktioner som experter inom området hos riksbanken, som trots att de själva sätter reporäntan har svårt att förutse dess utveckling vilket ni kan utläsa ur Figur 6. [9]

I data som är skapad för uppgifter kring tidsserieanalyser finns ofta mycket uppenbara trender, säsongsvariationer och sällan slumpartad variation men så är oftast inte fallet i

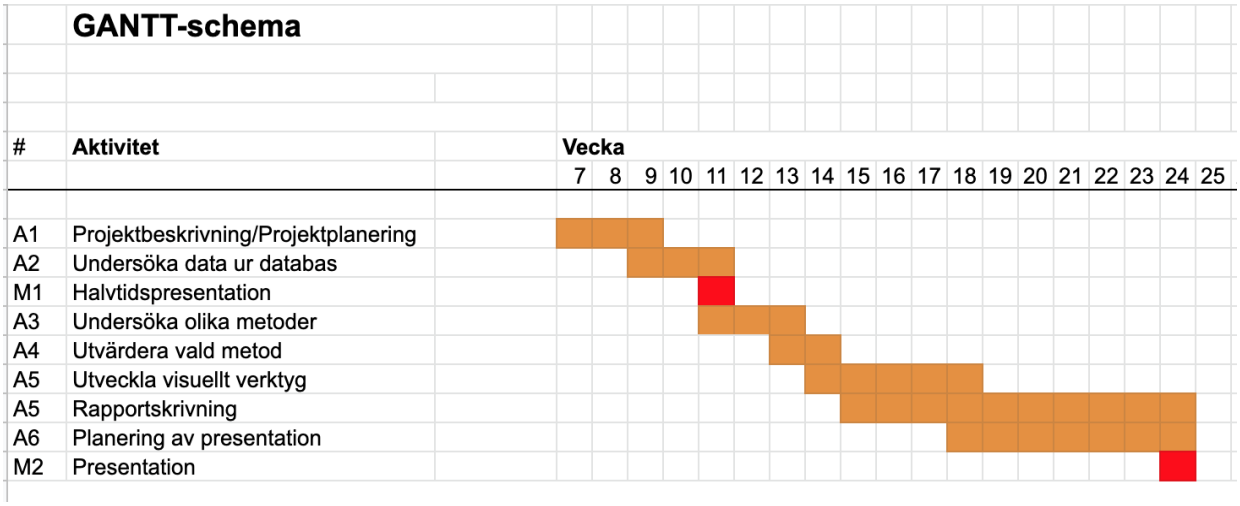
verkligheten. Om framtiden hade varit simpel att förutspå så hade alla blivit rika på börsen. Således bör man ta osäkerheten i beaktning när man nyttjar prognoserna vid stora beslut.[10]

3. Metodbeskrivning

3.1 Planering

I detta subkapitel kommer gruppens planeringsprocess, arbetssätt och vilka verktyg som nyttjats beskrivas.

3.1.1 Tidsplan



Figur 7. Visualisering av tidsplanen för projektet som övergripande beskriver vad som gjordes och när det utfördes

Under den inledande fasen så lades en ursprunglig tidsplan upp utifrån den diskussion som hade förts med företaget. Denna planering har förändrats över tid när allt mer insikt och informations har mottagits. Det har således funnits ett flertal olika iterationer av tidsplanen men i slutändan så var det denna tidsplan som följdes.

3.1.2 Trello

För hanteringen av de uppgifter som delades upp under projektets gång så nyttjades till viss del Trello för att få en visuell överblick kring vad som var klart och vad som skulle göras. Trello är ett samarbetsverktyg som underlättar vid organisering av projekt med hjälp av virtuella tavlor.[11]

3.1.3 Minimum Viable Product

Efter att den ursprungliga tidsplanen var färdig så kom projektgruppen överens om en MVP (Minimum Viable Product). Målet var att färdigställa åtminstone denna under projektets gång. I MVP:n ingick följande

- Hämtning av data från företagets ElasticSearch databas.
- Bearbetning av datan från databasen till en CSV fil.
- Implementation av API för generering av tidsserieanalyser och prediktioner.
- Visuell representation av prediktionen.
- Implementation av funktionaliteten i företagets befintliga analysverktyg

3.2 Kommunikation

För kommunikation kring arbetet med varandra likväl som de inblandade hos företaget så har kommunikationen sköts muntligt, via mejl samt via Slack.

3.2.1 Slack

Slack är en molnbaserad kommunikationsplattform som kan nyttjas över flera olika enheter eller plattformar. Den är menad för team och arbetsplatser, men kan också nyttjas för privata konversationer. Du kan ladda upp och dela filer samt integrera plattformen med många andra applikationer och tjänster som t.ex. Skype för videosamtal.[12]

3.3 Versionshantering

För versionshantering och integrering av den kod som utvecklades till den ursprungliga kodbasen så användes Git då det är vad företaget använder sig utav och även det som gruppen hade mest erfarenhet inom.
Git är ett aktivt underhållet projekt med öppen källkod som har hög mognadsgrad.
Det är i dagsläget det mest använda versionshanteringssystemet i världen och utvecklades ursprungligen av Linus Torvalds under 2005. Git fungerar väl på en stor del av alla operativsystem och i kombination med många olika integrerade utvecklingsmiljöer. [13]

3.4 Arbetsflöde

För att skapa ett effektivt arbetsflöde, där projektmedlemmar kunnat arbeta parallellt med olika funktioner och integrera dessa i analysverktyget smidigt så användes GitFlow [14]. Det går ut på att man utgår från en projektgren och utifrån den skapar man utvecklingsgrenar som blir kodbasen man arbetar mot. För varje ny funktion, snabbkorrigering eller refaktorering så skapas en ny gren från utvecklingsgrenen. Innan en ny gren tillåts integreras till utvecklingsgrenen så ska den testas och godkännas av en annan projektmedlem.

3.5 Testning

Testningen hölls väldigt simpel för projektet, om kod skulle pushas till utvecklingsgrenen så kollade den andra projektmedlemmen igenom koden och undersökte att den fungerade som det var tänkt, om så var fallet så var det tillåtet att pusha.

3.6 Designval

Då företaget hade en etablerad design i sitt analysverktyg så fanns inte oerhört mycket rum för kreativitet när det kom till design. Således försökte koden anpassas efter bästa förmåga för att överensstämma så bra som möjligt med deras befintliga design i analysverktyget.

3.7 Val av data

För att kunna utföra en tidsserieanalys så behövs data och då företaget besitter data kring ett flertal olika områden som berör deras produkter så var det fundamentalt att bestämma vilken data som skulle nyttjas för att generera en tidsserie. De besitter data kring installation av deras applikation, antal olika användare, antal installerade enheter och mycket mer. Vid diskussion med företaget så togs ett beslut om att det var specifikt den data kring installerade enheter som skulle nyttjas för tidsserieprediktionen då den ansågs vara vital för framtida beslutstaganden.

3.8 Utvecklingsverktyg

Detta kapitel ämnar att informera kring de utvecklingsverktyg som använts under projektet.

3.8.1 Visual Studio Code

För utveckling så har främst Visual Studio Code, även känt som VS Code nyttjats som integrerad utvecklingsmiljö. VS Code är en IDE som är utvecklad av Microsoft med öppen källkod och finns tillgänglig för Windows, MacOS och Linux. VS Code har bland annat stöd för avlusning, syntaxmarkering, intelligent kodavslutning samt Inbygda git kommandon. [15] Den huvudsakliga anledningen till att just denna IDE valdes var på grund av tidigare erfarenhet med den samt att den har stöd för de utvecklingsspråk som var planerade att användas för utvecklingen.

3.9 Utvecklingsspråk och bibliotek

Detta kapitel ämnar att informera kring de utvecklingsspråk och bibliotek som använts under projektets gång.

3.9.1 JavaScript

Det primärt använda språket i projektet var Javascript. JavaScript nyttjades för majoriteten av all utveckling under projektets gång.

JavaScript är ett scripting eller programmeringsspråk som bland annat tillåter utvecklare att implementera komplex funktionalitet på webbsidor. Varje gång en webbsida gör mer än enbart visar statisk information så kan du räkna med att JavaScript förmodligen är involverat. [16]

3.9.2 CSS

CSS (Cascading Style Sheets) nyttjades för styling av de utvecklade komponenterna.

CSS tillåter utvecklare att skapa visuellt tilltalande webbsidor. CSS är ett språk som specificerar hur dokument presenteras till användare, det vill säga hur de är stylade samt deras layout.[17]

3.9.3 HTML

HTML användes för att definiera viss struktur av innehåll i applikationen.

HTML är ett märkspråk som underlättar vid strukturering av innehåll. HTML består av en serie olika element som tillåter utvecklare att inkapsla olika delar av innehåll för att få det att presenteras eller agera på ett specifikt sätt.[18]

3.9.4 Python

Python nyttjades en del under den inledande fasen av projektet för att få en övergripande förståelse för hur tidsserieanalyser kan utföras. Språket nyttjades dels för testning av stationäritet med ett statistiskt test vid namn Augmented Dickey-Fuller test, [8] samt för en tidig visuell presentation av data med hjälp av bibliotek som pandas[19] och matplotlib[20]

Python är ett brett högnivåspråk som har en myriad av olika appliceringsområden. Språket är ett så kallat multi-paradigm språk[21] som stödjer procedurella, objektorienterade och vissa funktionella programmeringskonstruktioner. [22]

3.9.5 Node.js

Node.js är en open-source plattform som man använder sig av för att exekvera JavaScript kod utanför webbläsaren och är byggt på V8 JavaScript motorn. Node.js används ofta till att skapa API:er till webbapplikationer eller mobila applikationer.

Till skillnad från flera andra plattformar så körs en Node.js applikation enbart på en tråd och använder sig av icke-blockerande asynkrona I/O anrop. Detta gör Node.js snabbt, effektivt och skalbart. [23]

3.9.6 React

React är ett JavaScript bibliotek som används för att skapa användargränssnitt. En vanlig sammanfattning av React är att det representerar Vy-delen i en MVC-model, med andra ord de visuella delarna av ett program.

Grundidén med React är att man bygger sina applikationer med hjälp av återanvändbara komponenter. Att dela upp sina applikationer med React komponenter gör uppdateringarna effektiva, då React endast uppdaterar komponenter vars tillstånd har ändrats. React använder sig av "Document Object Model" (DOM) för att manipulera, webbsidors innehåll, struktur och design. [24]

3.9.7 BigML API

BigML har utvecklat ett API som kan implementeras med hjälp av olika programspråk och en av metoderna för att implementera det är genom ett bibliotek utvecklat för Node.js. Deras API gör det möjligt att använda all funktionalitet som deras webbapplikation tillhandahåller med några få metodanrop. [25]

3.9.8 Elasticsearch

Elasticsearch är en NoSQL-databas som företaget använder för att spara data från deras enheter. Elasticsearch har utvecklat ett bibliotek för Node.js som innehåller funktioner för att hämta och spara data i deras databas. [26]

3.9.9 Socket.IO

Socket.IO är ett JavaScript bibliotek som används för att skapa realtidsapplikationer. Biblioteket används för att underlätta vid kommunikation mellan webbklienter och servrar. Webbklienten i detta fall är skapad i React och servern i Node.js. [27]

3.9.10 D3

D3 står för datadrivna dokument och är ett JavaScript bibliotek som används för att skapa anpassade interaktiva datavisualiseringar. Det är ett bibliotek med öppen källkod som är skapat av Mike Bostock för att visualisera data i webbläsaren med hjälp av SVG, HTML och CSS.

SVG står för "Scalable Vector graphics" och nyttjas för att definiera vektorbaserad grafik i webbläsaren. Det definerar grafiken i XML-format där varje element och attribut i filen kan visualiseras. [28]

4. Undersökta metoder och API:er

Efter att den inledande planeringen av projektet var klar så lades majoriteten av tiden planenligt på att undersöka vilka befintliga metoder, API:er och verktyg som finns tillgängliga för att finna en lösning som passar företagets behov. I följande text följer en kort presentation utav de som ingått i vår undersökning och motivering kring varför BigML API valdes över de andra alternativen.

Inledningsvis undersöktes olika språk lämpliga för tidsserieanalys som exempelvis Python, olika varianter på olika metoder för tidsserieprediktioner så som Autoregressive Integrated Moving Average (ARIMA)[29], Holt-Winters Forecasting[30] samt hur data kan göras stationär så att den är lämpad för att nyttjas vid tidsserieprediktioner. Detta utfördes för att få en övergripande insikt i hur tidsserieanalyser och tidsserieprediktioner utförs, samt för att bättre kunna förstå oss på det API eller verktyg som senare skulle nyttjas för utvecklingen av analysverktyget hos företaget.

4.1 Cloud Machine Learning Engine

Är en AI-plattform utvecklad av Google som molnbaserat nyttjar kraften och flexibiliteten av ett flertal olika tekniker och verktyg. Som användare kan du nyttja denna motor för att träna modeller för maskininlärning med hjälp av resurser från Google Cloud Platform. Anledningen till denna plattform inte valdes är för att det fanns snabbare alternativ som framstod som lättare att integrera [32], dessutom hade ingen i projektgruppen erfarenheter av någon av teknikerna för Googles plattform eller maskininlärning, således kändes utmaningen inte rimlig under projektets tidsram. [31]

4.2 Amazon Forecast

En tjänst utvecklad av Amazon som nyttjar maskininlärning för att skapa mycket tillförlitliga tidsserieprediktioner. Som användare av tjänsten behöver en inte tänka på maskininlärning, serverhantering och användaren behöver enbart betala för det som hen nyttjar. Tyvärr så är denna tjänst fortfarande i ett tidigt teststadie och enbart tillgänglig i utvalda stater i USA. Om Amazon Forecast hade varit en färdig tjänst och tillgänglig i Sverige så hade den kunnat vara ett mycket intressant alternativ. På grund av bristen på tillgänglighet och att tjänsten fortfarande är under utveckling så undersöktes den inte ytterligare. [33] [34]

4.3 BigML API

BigML är en tjänst för molnbaserad hosting för maskininlärning och dataanalys. Användare kan enkelt inkludera en källa för uppladdning av data, skapa dataset av datan, skapa en modell baserad på datasetet och därefter bilda prediktioner utifrån modellen. [35]

BigML API tillåter användare att få tillgång till BigML så att de kan integrera dess funktionalitet i andra applikationer. Anledningen till att detta API valdes är på grund av följande punkter

- BigML API går att nyttja för integrering i en applikation via Node.js vilket är vad företaget nyttjar för back-end.
- Det gör det möjligt för utvecklare utan erfarenhet av maskininlärning och tidsserieprediktioner att enkelt skapa tillförlitliga prediktioner, som testar flera olika metoder för tidsserieprediktioner med enkla metodanrop.

- Det framstod utan tvekan som den snabbaste av de befintliga lösningar som har involverats i undersökningen. [32]

4.3.1 Generering av tidsserieprediktion

Då prediktioner baserade på företagets data ej kan delas av rättsliga skäl så kommer exempel baserad på data ämnad för denna typ av uppgift nyttjas för att ge en överblick kring hur våra prediktioner skapas.

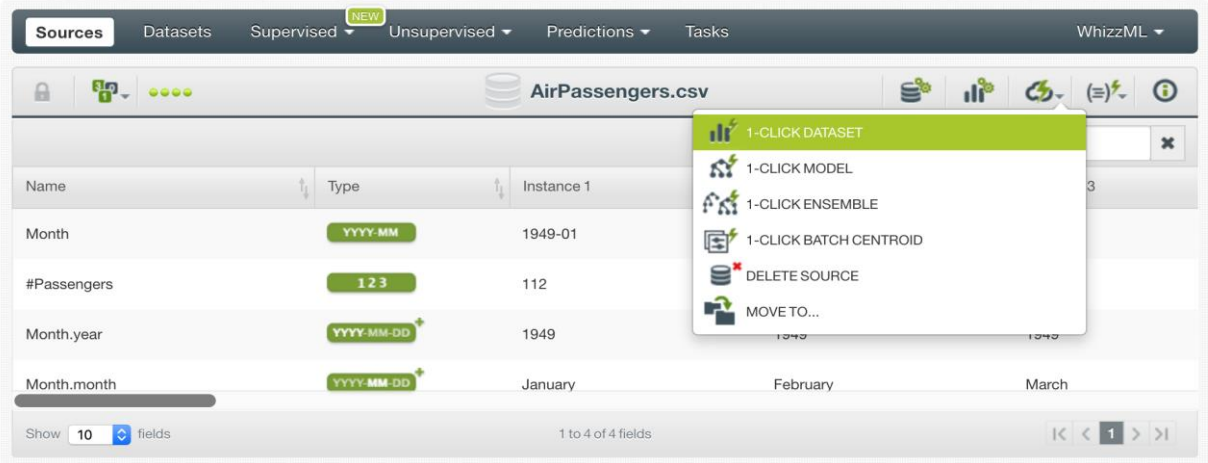
I resterande subkapitel till kapitel 4.3 kommer ett exempel på hur en tidsserieprediktion baserad på data kring antalet passagerare som rest med flyg månatligen bildas.[44] Tidsserien nyttjar datan för antalet passagerare för att bilda prediktionen och månaden som index. I detta exempel kommer BigML's plattform nyttjas för visualisering av prediktionen.
Det en användare behöver för att kunna replikera detta scenario är en CSV fil med numeriska värden samt ett konto hos BigML.[36][37]

4.3.2 Uppladdning av data

För att kunna generera en tidsserie behövs en källfil med data att basera tidsserien på. Således behöver användaren inledningsvis ladda upp en källfil under source inne på BigML's plattform vilket kan göras antingen genom dra och släpp funktionaliteten eller genom knappen visad i bilden ovan. När datan laddats upp så kommer datatyperna av fälten i filen automatiskt detekteras. Enbart de fält med numeriska värden kommer nyttjas. [38]

Viktigt att notera är att indexen i BigML kommer vara i samma ordning som de är i den ursprungliga källfilen, dvs att den första datapunkten i tidsserien kommer vara den första raden i källfilen och således är det viktigt att se till så att raderna i källfilen är ordnade i kronologisk ordning.

4.3.3 Generering av dataset



Figur 8. Vy från BigML's plattform med knapp för generering av dataset markerad

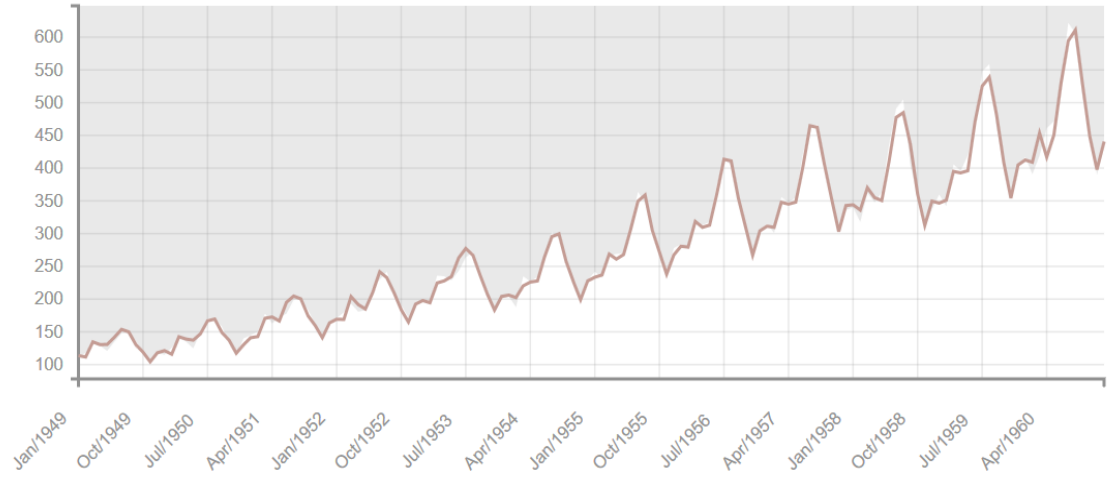
Från vyn för den uppladdade källan så kan ett dataset genereras från menyn som är visualiserad i bilden ovan. Detta resulterar i en strukturerad version av datan som bland annat kan nyttjas för att skapa en tidsserie.

4.3.4 Generering av tidsserie

Figur 9. Vy från BigML's plattform med knapp för generering av tidsserie markerad

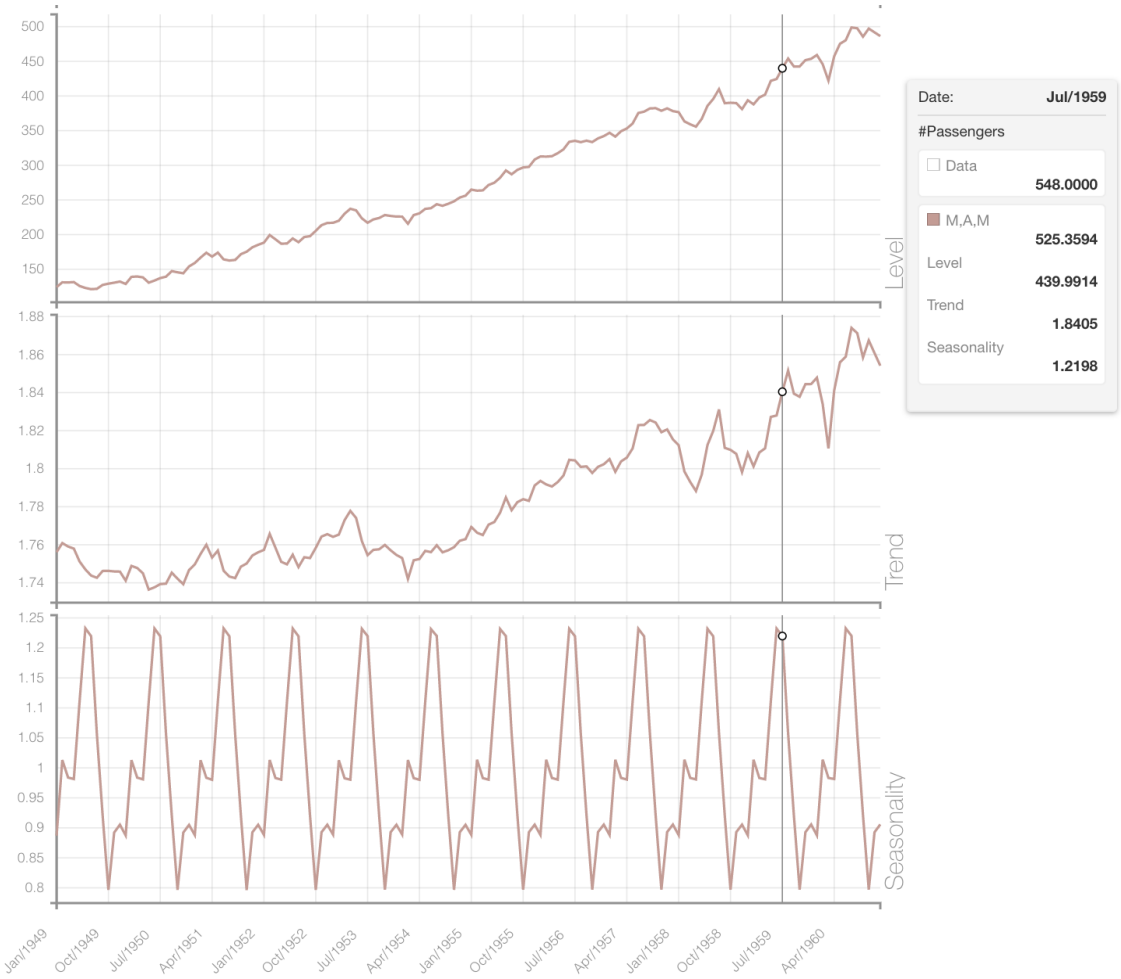
Efter att ett dataset är skapat utifrån källfilen så kan en tidsserie skapas genom menyn som är visualiserad i bilden ovan. Det går även att manuellt konfigurera olika parametrar som kan påverka resultaten av tidsserien men dessa inställningar är något som ej kommer redovisas här.

4.3.5 Utvärdering av tidsserie



Figur 10. Vy från BigML's med visualisering av tidsserie

Efter att ett tidsserien är genererad utifrån datasetet så möts användaren i dess vy av en visuell representation av datasetet tillsammans med en vald modell. Denna vy kan användare bland annat nyttja för att generera en nedbruten version av modellen för tidsserien.

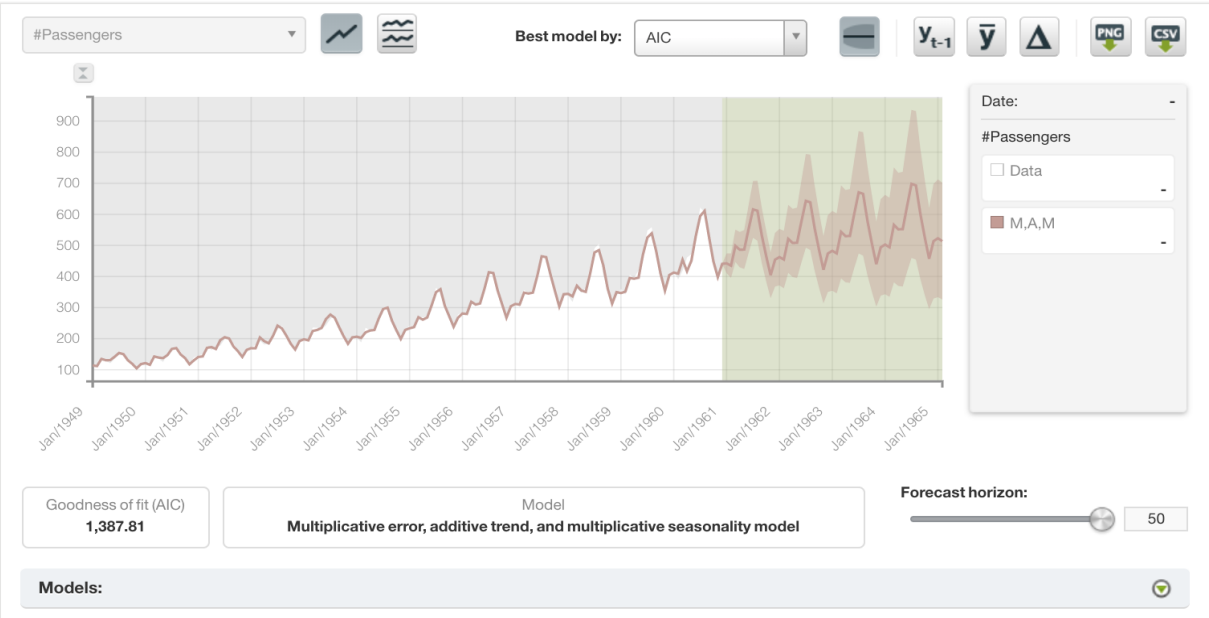


Figur 11. Vy från BigML's plattform med visualisering av nedbruten modell för tidsserie

Från vyn med den nedbrutna modellen så kan användaren utläsa information om nivån, trenden och säsongsvariationen för modellen. (Se kapitel 2.1.1) Dessa är de faktorer som tillsammans med den valda modellen utgör prediktionen för tidsserien. (Se kapitel 2.1.1)

Modellen väljs genom att API:et testar alla de modeller som den kan tillämpa på datasetet och undersöker vilken modell som bäst överensstämde med verkligheten. Undersökningen går till så att BigML som standardinställning nyttjar 80% av datan för träning av modellerna och 20% för testning av dem. Den modell som under testningsfasen låg närmast de verkliga värdena är den som BigML returnerar.

4.3.6 Visualisering av tidsserieprediktion



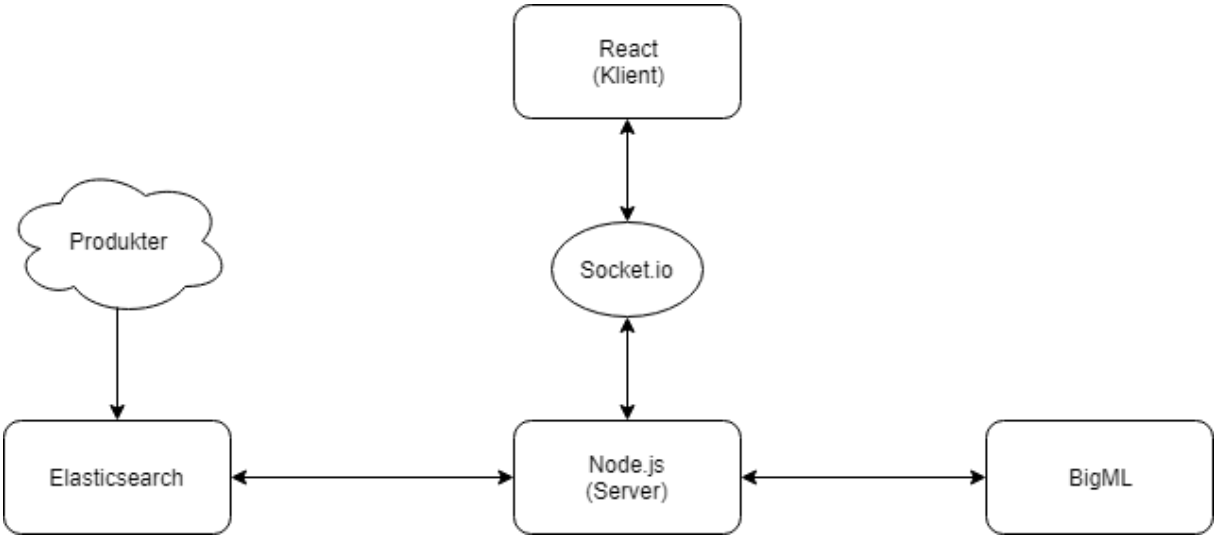
Figur 12. Vy från BigML's plattform med visualiserad tidsserieprediktion

Prediktionen för tidsserien kan visualiseras genom att ändra reglaget "Forecast horizon" i den ursprungliga vyn för tidsserien.

5. Konstruktion

Företaget önskade en implementation direkt in i deras befintliga analysverktyg som de skapat, vilket innebar att en stor kodbas redan fanns för både backend och frontend. Deras analysverktyg innehöll redan en visualisering av data från deras databas i stapeldiagram, vilket liknade det önskvärda syftet vårt projekt ämnade att realisera. Detta resulterade i att majoriteten av de olika programspråk och bibliotek som nyttjades för projektet redan var etablerade i den befintliga kodbasen hos företaget. Det enda bibliotek som nyttjades som inte var en del av deras befintliga kodbas var BigML API. Hur de API:er, programspråk och bibliotek som användes i konstruktionen har nyttjats och hur de sedan kommunicerade med varandra för att skapa ett verktyg som kan prognostisera en tidsserie presenteras i detta kapitel.

5.1 Systemkonstruktion



Figur 13. Flödesschema för systemkonstruktionen

När företagets enheter installeras sparas data i en Elasticsearch databas med fält för datum och vilken produkt det handlar om. De har sparat data i denna databas sedan slutet av år 2016. Första steget i systemkonstruktionen var att hämta all data för installerade enheter genom att använda Elasticsearch biblioteket i Node.js. Biblioteket har funktionalitet för att göra queries mot databasen och denna funktionalitet används för att hämta all dagsdata för installerade enheter(Se bilaga 1).

Nästa steg i systemet är att använda BigML för att generera en prediktion baserad på den historiska datan. Detta utfördes med hjälp av BigML´s bibliotek för Node.js som tillhandahåller funktionalitet för att använda sig av alla funktioner från BigML som visat i

1. Skapa en källa för datan som ska prognostiseras
2. Skapa ett dataset från källan
3. Skapa en tidsserie baserad på datasetet
4. Skapa en prediktion på tidsserien

BigML tar endast filer som indata och således behövs datan från Elasticsearch göras om till en fil då vår databasbegäran ger oss den råa datan. Den råa datan skickas till en CSV-parser som skriver en CSV-fil med den historiska datan(Se bilaga 2).

När den historiska datan har konverterats till en CSV-fil kan processen för prediktion av datan påbörjas. Filen skickas som indata till BigML's source funktion, som skapar en källa för datan. Källan används i sin tur för att skapa ett dataset som innehåller två fält, ett för datum och ett som innehåller antalet installationer för respektive datum(Se bilaga 3). Datasetet skickas därefter som indata för att skapa en tidsserie som analyserar vilka trender, nivåer eller variationer datan har. Till slut anropas BigML's forecast funktion som tar tidsserien som indata och returnerar med standardinställningen 50 prognostiserade punkter med en övre gräns och lägre gräns som representerar felmarginalen för prediktionen. För varje gång man skapar en tidsserie, dataset eller källa så sparas dessa hos BigML och för att inte övertrassera den gratis data man får tillgång till hos BigML så tas dessa bort så fort prediktionen är klar(Se bilaga 4).

När BigML har returnerat de prognostiserade värdena så är all relevant data hämtad, således är det tid för datan att skickas från servern till klienten. Men innan datan kan skickas bör den sorteras och delas in i fält för att underlätta arbetet med datan i klienten. Alla punkter sorteras in i objekt som innehåller fält för datum, värde och en boolean som håller koll på om värdet är ett prognostiserat värde eller ej. De prognostiserade värdena delas också upp i objekt som innehåller fält för det prognostiserade värdet, övre gränsen och undre gränsen. Objekten som representerar prognostiserade värden anges som värden för objekten som representerar punkter(Se bilaga 5).

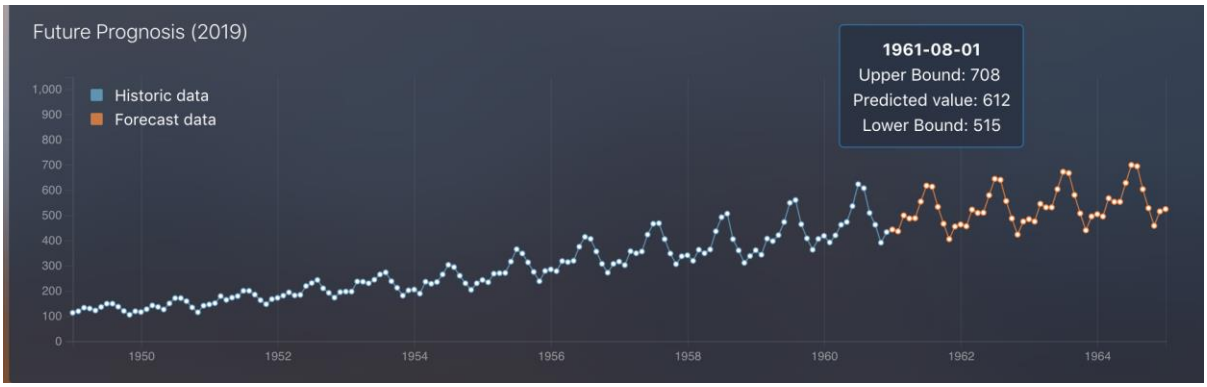
När datan är sorterad ska den skickas till klienten via kommunikationen mellan server och klient. Denna kommunikation är konstruerad med hjälp av biblioteket Socket.IO. Detta bibliotek innehåller funktionalitet för anrättning av struktur för websockets som i applikationen nyttjas genom att klienten lyssnar kontinuerligt på vad webb-socketen returnerar för att hålla sig uppdaterad. Det befintliga analysverktyget innehöll redan funktionalitet för webb-socket som användes för denna konstruktion.

Klienten är uppbyggd med React komponenter, och varje komponent har metoder som exekveras när en komponent monteras i webbläsaren eller när den avmonteras. Dessa metoder nyttjas så att varje gång komponenten som innehåller all funktionalitet för visualisering av grafen renderas, anrättas en anslutning mellan server och klient. Servern skickar då den data som efterfrågats via webb-socketen som då kan behandlas på klientsidan för att visualiseras i grafen. När komponenten avmonteras stängs webb-socketen och öppnas först igen när komponenten monteras på nytt.

För datavisualisering i klienten användes biblioteket D3.js för React då det var de biblioteket som företaget hade valt att nyttja för all datavisualisering i deras analysverktyg. D3.js innehåller funktionalitet för att addera, manipulera eller ändra HTML element. För att visualisera datan som skickas från servern, skapas först ett SVG-element som representerar hela grafen. I detta element adderas en x-axel för representation av tid, en y-axel som representerar värdet och ett rutnät för att simplifiera visualiseringen av punkter. Efter detta så adderas ett line-element som manipuleras att vara i blå färg för historisk data och orange för prognostiserad data. Sista steget i visualiseringen är att skapa circle-element som representerar punkter, och när man håller musen över dessa får man värdet och datumet för punkten.

Figur 14. Visualisering av tidsserieprediktion i företagets analysverktyg

Figur 15. Visualisering av tidsserieprediktion med historisk punkt markerad

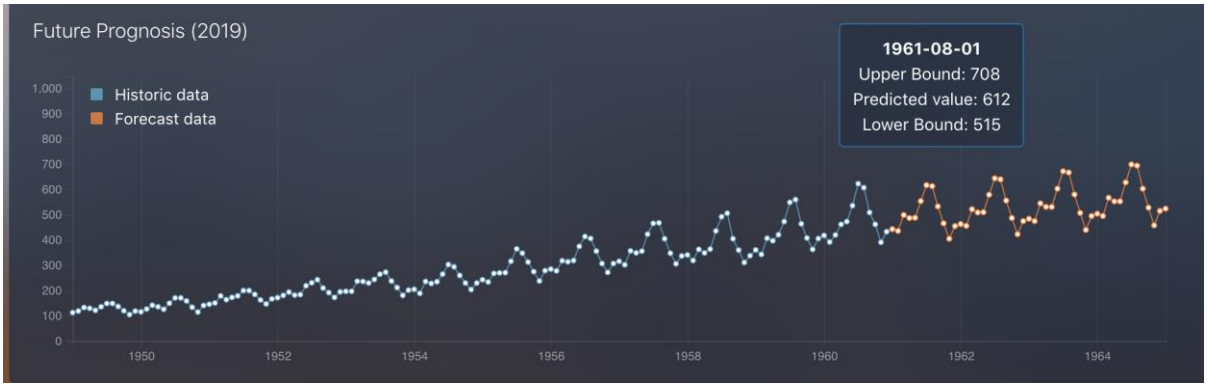


Figur 16. Visualisering av tidsserieprediktion med prognostiserad punkt markerad.

6. Resultat

Målet var att finna ett verktyg eller API som tillhandahåller funktionalitet kring generering av prediktioner utifrån en tidsserie. Det var vitalt att lösningen skulle kunna implementeras i Node.js eller React för att det skulle kunna integreras i företagets analysverktyg. Efter undersökning av olika metoder och API:er valdes BigML API. Detta API implementerades in i företagets befintliga verktyg.

Efter att ett API valts ut och integrerats kom gruppen överens om en MVP(Se kapitel 3.1.3) som konstruerades inom projektets tidsram. Företaget önskade en tydlig visualisering av tidigare data samt prognostiserad data som designmässigt skulle överensstämma med deras befintliga verktyg.



Figur 17. Visualisering av tidsserieprediktion med historisk punkt markerad

I figuren ovan så kan man utläsa resultat kring visualiseringen av datan, den historiska datan är visualiserad med en blå färg medan den prognostiserade datan har en orange färg.

Prediktioner skapades under projektets gång på exempeldata såväl som företagets riktiga data. Resultaten för prediktionerna på exempeldatan visade ofta en väldigt stor säkerhet i form av ett lågt konfidensintervall, medan prediktionerna på företagets data resulterade i ett större konfidensintervall och därmed mindre pålitliga resultat. BigML lyckades inte identifiera en trend på företagets data vilket resulterade i en horisontell kurva.

Efter färdigställande av MVP:n fanns tid över för städning och kommentering av kod, finslipning av design, test med annan data baserad på en uppgift gjord för tidsserieanalys och reflektion kring vad som hade kunnat göras annorlunda om projektet hade replikerats.

7. Slutsatser / Analyser och Diskussioner

En stor del av besluten kring vilka bibliotek och utvecklingspråk som skulle nyttjas var redan etablerade då företaget önskade en implementation i deras befintliga verktyg som redan hade en stor kodbas. Detta resulterade i att gruppen fick spendera mer tid än önskat på att lära sig vissa komplexa bibliotek som hellre hade lagts på att skapa mer funktionalitet och tydligare visualisering av konfidensintervallet.

Ett av de primära besluten som togs var vilken metod som skulle nyttjas för att generera tidsserieprediktioner, där valet föll på BigML's API(Se sektion 4.3). Om någon av de andra metoderna hade valts, skulle förmodligen gruppen inte hunnit implementera en lösning direkt in i företagets verktyg. Detta då gruppen saknade förkunskaper inom områden som var viktiga för dessa metoder, som maskininlärning och matematisk statistik. I bästa fall hade gruppen hunnit skapa en tidsserieprediktion baserat på en enda modell, medans BigML baserar sin tidsserieprediktion på många olika modeller.

BigML som gruppen valde ut att arbeta med fungerade väldigt bra, även om det var bristfälligt med dokumentation och det tog längre tid än väntat att implementera. De fördelar som BigML hade över andra metoder som hade undersökts var dels att det testade en mängd olika modeller för generering av tidsserieprediktioner samt att det fanns stöd för implementering i Node.js.

Den huvudsakliga sluttsatsen av projektet är att framtiden är oviss och att prediktioner kring framtida data ofta präglas av hög osäkerhet(Se sektion 2.1.5). När exempeldata samlad för uppgifter kring tidsserieprediktioner nyttjas så är trenderna och säsongsvariationerna oftast tydliga medan verklig data ofta präglas av fler slumpartade element. Trots att osäkerhetsnivån kring prediktionerna är hög så anser gruppen oss ha uppfyllt vårt syfte då vår MVP blev färdigställd. Syftet var att integrera en vy i företaget analysverktyg kring tidsserieprediktioner för att underlätta vid produktionsrelaterade beslut. Målet verkställdes med hjälp av BigML's API och delar ur deras befintliga kodbas.

7.1 Utmaningar

Under projektets gång stötte gruppen på en del utmaningar som längre tid än väntat att lösa. I detta kapitel kan du läsa om några av projektets större utmaningar.

7.1.1 Sätta sig in i en etablerad kodbas

En av de primära utmaningarna med projektet var att sätta sig in i företagets kod för det existerande analysverktyget då dokumentationen för det var nästintill obefintlig och koden till viss del var i behov av ytterligare kommentarer för att utomstående enklare ska kunna få en förståelse för den. Bristen på dokumentation resulterade i att mycket tid investerades för att få en övergripande förståelse kring flödet för applikationen. Att sätta sig in i en del av de bibliotek som de nyttjade för applikationen var även en del av denna utmaning då bland annat D3.js var ett bibliotek vars komplexitet resulterade i att utvecklingen av den visuella representation tog mer tid än vad som var önskvärt.

7.1.2 Förståelse kring tidsserieanalyser och tidsserieprediktioner

Den andra huvudsakliga utmaningen var att få en övergripande insikt i hur tidsserieanalyser fungerar och hur tidsserieprediktioner kan utföras. Denna utmaning krävde omfattande

läsning av olika artiklar, utdrag ur böcker och guider om ämnet då gruppmedlemmarna sedan tidigare hade nästintill obefintlig kunskap om ämnet.

7.1.3 Fastställa syfte

En av utmaningarna med projektet var att få en konkret bild kring vad det var som skulle utvecklas.
Vid den inledande kommunikationen med företaget så behandlade diskussionen mycket som skulle kunna göras men inga konkreta beslut togs. Det var således svårt att till en början få en övergripande förståelse kring vad som då var syftet med projektet. Tack vare vår handledare Koen Claessen så fastställdes ett mer konkret syfte efter diskussion med honom som vid dialog med företaget mottogs väl.

7.1.4 Få tillgång till utvecklingsrelaterade förnödenheter

En annan utmaning var att få tillgång till det som behövdes för att kunna inleda utvecklandet av funktionaliteten i deras analysverktyg. Att det tog sådan tid innan tillgång mottogs för de licenser, nycklar och liknande var något som inledningsvis bromsade projektet.

7.1.5 Begränsad dokumentation av BigML's Node.js bibliotek

Dokumentationen för BigML's API var mest detaljerad för HTTP requests och Python, samt att det fanns mycket exempel på hur man kan använda de olika funktionerna. Det fanns även dokumentation för Node.js men dessvärre inte lika detaljerad och bara ett fåtal exempel. Detta resulterade i att små problem ofta tog längre tid än vad det borde göra, och ofta fick BigML's support kontaktas för att få svar på hur våra problem kunde lösas.

7.2 Återkoppling från företag

Överlag så var företaget nöjda med implementationen, det de hade önskat var en tydligare visuell representation av konfidensintervallet genom att rita ut det istället för att ha med värdena i dialogrutan för punkterna.

7.3 Förslag till fortsatt arbete

Om fortsatt utveckling av funktionaliteten vore relevant så hade det varit intressant att implementera olika metoder och verktyg för generering av tidsserieprediktioner. Efter implementation av de olika metoderna så hade man därefter kunnat jämföra resultaten av de olika metoderna för att undersöka om BigML var rätt väg att gå eller om det fanns andra verktyg som hade varit mer lämpliga ur vissa perspektiv att nyttja för uppgiften. Rent utvecklingsmässigt så hade det förmodligen varit fördelaktig att även visualisera konfidensintervallet i grafen för att göra osäkerheten i tidsserieprediktionen tydligare.

7.4 Etik

Detta projekt kretsar till stor del kring nyttjandet av insamlad data vilket kan anses vara en relevant etisk aspekt kring projektet. Under projektet samlas data in från en databas som innehåller personuppgifter, men gruppen hämtar enbart data som berör antal dagliga installationer. Inte under något något skede i projektet som gruppen ansvarar över har känslig data nyttjats.

7.5 Avslutande ord

Överlag är gruppen nöjd med arbetsinsatsen och slutresultatet med tanke på hur omfattande och utmanande åtagandet kring projektet var för samtliga medlemmar. Båda har kunnat dra stor lärdom från de motgångar som infann sig under projektets gång samt kring problematiken att jobba med nya verktyg och i en väl etablerad kodbas med bristfällig dokumentation.

8. Referenser och Bilagor

8.1 Referenser

[1] J.Holst, "Nationalencyklopedin, tidsserieanalys"[online]. Tillgänglig: https://www.ne.se/uppslagsverk/encyklopedi/l%C3%A5ng/tidsserieanalys [Accessed 30 Apr. 2019].
[2] A.Brasetvik, " Elasticsearch as a NoSQL Database"[online]. Tillgänglig: https://www.elastic.co/blog/found-elasticsearch-as-nosql Accessed 30 Apr. 2019].
[3] Ekonomifakta. (2009). *Tidsserie - Ekonomifakta*. [online] Available at: https://www.ekonomifakta.se/Ordlista/Tidsserie/ [Accessed 30 Apr. 2019].
[4] Slättman, H. (2018). *Tidsserier - Statistiska Institutionen*. [online] Statistics.su.se. Available at: https://www.statistics.su.se/forskning/tidsserier [Accessed 30 Apr. 2019].
[5] Harder, D., Rodgers, J. and (jrodgers@admmail.uwaterloo.ca), D. (2005). *Topic 6.4: Extrapolation*. [online] Ece.uwaterloo.ca. Available at: https://ece.uwaterloo.ca/~dwharder/NumericalAnalysis/06LeastSquares/extrapolation/complete.html [Accessed 30 Apr. 2019].
[6] Jesus, M. (2017). *Introduction to Time Series*. [online] The Official Blog of BigML.com. Available at: https://blog.bigml.com/2017/07/12/introduction-to-time-series/ [Accessed 30 Apr. 2019].
[7] Singh, A. (2018). *A Gentle Introduction to Handling a Non-Stationary Time Series in Python*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2018/09/non-stationary-time-series-python/ [Accessed 30 Apr. 2019].
[8] Brownlee, J. (2016). *How to Check if Time Series Data is Stationary with Python*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/time-series-data-stationary-python/ [Accessed 30 Apr. 2019].
[9] Cornucopia? - Evig tillväxt i en ändlig värld?. (2016). *Riksbanken tror alltid på högre räntor*. [online] Available at: https://cornucopia.cornubot.se/2016/05/riksbanken-tror-alltid-pa-hogre-rantor.html [Accessed 30 Apr. 2019].
[10] Researchoptimus.com. (2019). *What is Time Series analysis?*. [online] Available at: https://www.researchoptimus.com/article/what-is-time-series-analysis.php [Accessed 30 Apr. 2019].
[11] Help.trello.com. (2019). *What is Trello? - Trello Help*. [online] Available at: https://help.trello.com/article/708-what-is-trello [Accessed 30 Apr. 2019].
[12] Pocket-lint.com. (2019). *Slack*. [online] Available at: https://www.pocket-lint.com/apps/news/136472-what-is-slack-and-how-does-it-work [Accessed 30 Apr. 2019].
[13] Atlassian. (2019). *What is Git: become a pro at Git with this guide | Atlassian Git Tutorial*. [online] Available at: https://www.atlassian.com/git/tutorials/what-is-git [Accessed 30 Apr. 2019].
[14] Atlassian. (2019). *Gitflow Workflow | Atlassian Git Tutorial*. [online] Available at: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow [Accessed 9 Jan. 2019].
[15] Code.visualstudio.com. (2019). *Visual Studio Code - Code Editing. Redefined*. [online] Available at: https://code.visualstudio.com/ [Accessed 30 Apr. 2019].
[16] MDN Web Docs. (2019). *What is JavaScript?*. [online] Available at: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript [Accessed 30 Apr. 2019].
[17] MDN Web Docs. (2019). *CSS: Cascading Style Sheets*. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/CSS [Accessed 30 Apr. 2019].

[18] MDN Web Docs. (2019). *HTML: Hypertext Markup Language*. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/HTML [Accessed 30 Apr. 2019].
[19] Pandas.pydata.org. (2019). *Python Data Analysis Library — pandas: Python Data Analysis Library*. [online] Available at: https://pandas.pydata.org [Accessed 30 Apr. 2019].
[20] Matplotlib.org. (2019). *Matplotlib: Python plotting — Matplotlib 3.0.3 documentation*. [online] Available at: https://matplotlib.org/ [Accessed 30 Apr. 2019].
[21] Sv.wikipedia.org. (2019). *Programmeringsparadigm*. [online] Available at: https://sv.wikipedia.org/wiki/Programmeringsparadigm [Accessed 30 Apr. 2019].
[22] MDN Web Docs. (2019). *Python*. [online] Available at: https://developer.mozilla.org/en-US/docs/Glossary/Python [Accessed 30 Apr. 2019].
[23] Introduction to Node.js. (2019). *Introduction to Node.js*. [online] Available at: https://nodejs.dev/ [Accessed 30 Apr. 2019].
[24] Reactjs.org. (2019). *React – A JavaScript library for building user interfaces*. [online] Available at: https://reactjs.org/ [Accessed 30 Apr. 2019].
[25] BigML.com - Machine Learning made easy. (2019). *BigML.com*. [online] Available at: https://bigml.com/ [Accessed 30 Apr. 2019].
[26] Elastic.co. (2019). *Open Source Search & Analytics · Elasticsearch | Elastic*. [online] Available at: https://www.elastic.co/ [Accessed 30 Apr. 2019].
[27] Socket.IO. (2019). *Socket.IO*. [online] Available at: https://socket.io/ [Accessed 30 Apr. 2019].
[28] Bostock, M. (2019). *D3.js - Data-Driven Documents*. [online] D3js.org. Available at: https://d3js.org/ [Accessed 30 Apr. 2019].
[29] Investopedia. (2019). *Autoregressive Integrated Moving Average (ARIMA)*. [online] Available at: https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp [Accessed 30 Apr. 2019].
[30] Jinka, P. (2019). *Holt-Winters Forecasting Simplified*. [online] Vividcortex.com. Available at: https://www.vividcortex.com/blog/holt-winters-forecasting-simplified [Accessed 30 Apr. 2019].
[31] Google Cloud. (2019). *AI Platform documentation | AI Platform | Google Cloud*. [online] Available at: https://cloud.google.com/ml-engine/docs/ [Accessed 30 Apr. 2019].
[32] Dorard, L. (2015). *Machine Learning Wars: Amazon vs Google vs BigML vs PredicSis*. [online] Kdnuggets.com. Available at: https://www.kdnuggets.com/2015/05/machine-learning-wars-amazon-google-bigml-predicsis.html [Accessed 30 Apr. 2019].
[33] Amazon Web Services, Inc. (2019). *Time Series Forecasting | Machine Learning | Amazon Forecast*. [online] Available at: https://aws.amazon.com/forecast/ [Accessed 30 Apr. 2019].
[34] Amazon Web Services, Inc. (2019). *Introducing Amazon Forecast – Now in Preview*. [online] Available at: https://aws.amazon.com/about-aws/whats-new/2018/11/introducing-amazon-forecast-now-in-preview/ [Accessed 30 Apr. 2019].
[35] BigML.com - Machine Learning made easy. (2019). *Overview | BigML.com API*. [online] Available at: https://bigml.com/api/overview [Accessed 30 Apr. 2019].
[36] BigML.com - Machine Learning made easy. (2019). *Impressed by ease of use of BigML*. [online] Available at: https://bigml.com/accounts/register/ [Accessed 30 Apr. 2019].
[37] Kaggle.com. (2019). *Explore: flights.csv | airports.csv | airlines.csv | Kaggle*. [online] Available at: https://www.kaggle.com/miquar/explore-flights-csv-airports-csv-airlines-csv [Accessed 30 Apr. 2019].
[38] BigML.com - Machine Learning made easy. (2019). *BigML is Machine Learning made easy*. [online] Available at: https://bigml.com/dashboard/sources [Accessed 30 Apr. 2019].
[39] Cornucopia? - Evig tillväxt i en ändlig värld?. (2016). *Riksbanken tror alltid på högre räntor*. [online] Available at: https://cornucopia.cornubot.se/2016/05/riksbanken-tror-alltid-

pa-hogre-rantor.html [Accessed 20 May 2019].
[40]: Statistics How To. (2018). Exponential Smoothing: Definition of Simple, Double and Triple - Statistics How To. [online] Available at: https://www.statisticshowto.datasciencecentral.com/exponential-smoothing/ [Accessed 11 Jun. 2019].
[41]: Gellerstedt, M. (2013). *Grunderna i SPSS*. [online] Hv.se. Available at: https://www.hv.se/globalassets/dokument/utbilda/intro-till-spss-20.pdf [Accessed 11 Jun. 2019].
[42]: En.wikipedia.org. (2019). Stationary process. [online] Available at: https://en.wikipedia.org/wiki/Stationary_process#/media/File:Stationarycomparison.png [Accessed 12 Jun. 2019].
[43]: En.wikipedia.org. (2019). *Stationary process*. [online] Available at: https://en.wikipedia.org/wiki/Stationary_process [Accessed 11 Jun. 2019].
[44]: Community.qlik.com. (2017). AirPassengers.csv. [online] Available at: https://community.qlik.com/t5/Qlik-Server-Side-Extensions/AirPassengers-csv/ta-p/1476807 [Accessed 12 Jun. 2019].

8.2 Bilagor

Bilaga 1:

```
60          let params = {
61              query: {
62                  bool: {
63                      must: { match: { counted: true } },
64                      filter: {
65                          range: {
66                              createdAt: {
67                                  gte: `2016`,
68                                  lte: 'now/y+1y/y',
69                                  time_zone: 'Europe/Stockholm'
70                              }
71                          }
72                      }
73                  }
74              },
75              aggregations: {
76                  sum_days: {
77                      date_histogram: {
78                          field: 'createdAt',
79                          interval: '1d',
80                          min_doc_count: 0,
81                          format: 'yyyy-M-d',
82                          time_zone: 'Europe/Stockholm',
83                      }
84                  }
85              }
86          }
87
88          try {
89
90              self.searching = true;
91
92              const response = await elasticsearch.Query(self.indexName, params);
```

Bilaga 2:

```
92          const response = await elasticsearch.Query(self.indexName, params);
93          const sum_days = response.aggregations.sum_days.buckets;
94
95          // Remove values from current day
96          sum_days.pop();
97
98          // Sort all historic values to be a correct CSV which can be read by BigML API as a source
99          let csvData = "Date,Value";
100         sum_days.forEach((d) => {
101             if (d) {
102                 csvData = csvData.concat("\n" + d.key_as_string + ',' + d.doc_count);
103             }
104         });
105
106         // Write a csv file which is used as source for forecast
107         fs.writeFileSync('installations.csv', csvData, 'utf8');
108
109         // Get forecast from BigML API
110         let forecastData = await bigml.getForecast();
```

Bilaga 3:

```
// Creates a source with data from a file
function createSource() {
    const source = new bigml.Source(connection);
    const data = 'installations.csv';
    const promise = new Promise((resolve, reject) => {
        return source.create(
            data
        , (error, response) => {
            if (error) {
                return reject(error);
            }
            if (response) {
                return resolve(response);
            }
        });
    });
    return promise;
}

// Creates a dataset from a source
function createDataset(source) {
    const dataset = new bigml.Dataset(connection);
    const promise = new Promise((resolve, reject) => {
        return dataset.create(
            source
        , (error, response) => {
            if (error) {
                return reject(error);
            }
            if (response) {
                return resolve(response);
            }
        });
    });
    return promise;
}
```

Bilaga 4:

```
// Creates a timeseries from a dataset
function createTimeseries(dataset) {
    const timeseries = new bigml.TimeSeries(connection);
    const promise = new Promise((resolve, reject) => {
        return timeseries.create(
            dataset
        , (error, response) => {
            if (error) {
                return reject(error);
            }
            if (response) {
                return resolve(response);
            }
        });
    });
    return promise;
}

// Creates a forecast from a timeseries
function createForecast(timeseries) {
    const forecast = new bigml.Forecast(connection);
    const promise = new Promise((resolve, reject) => {
        return forecast.create(
            timeseries,
            { "Value": { "horizon": 50 } }
        , (error, response) => {
            if (error) {
                return reject(error);
            }
            if (response) {
                return resolve(response);
            }
        });
    });
    return promise;
}
// Creates a forecast and returns the values. Deletes all resources when done
const getForecast = async () => {
    const source = await createSource();
    const dataset = await createDataset(source);
    const timeseries = await createTimeseries(dataset);
    const forecast = await createForecast(timeseries);
    deleteSource(source.resource);
    deleteDataset(dataset.resource);
    deleteTimeSeries(timeseries.resource);
    return forecast;
}
```

Bilaga 5:

```
        self.forecastObject = {
            point_forecast: "",
            lower_bound: "",
            upper_bound: "",
        }

        self.dayObject = {
            date: "",
            value: "",
            forecast: false
        }

    getSocketData(historicData, forecastData) {
        const self = this;
        let sortedData = []
        // Parse historic data to dayData objects and add to sortedData
        historicData.forEach((d) => {
            if (d) {
                let [year, month, day] = d.key_as_string.split('-');
                let tempDate = DateTime.fromObject({
                    year: parseInt(year, 10),
                    month: parseInt(month, 10),
                    day: parseInt(day, 10),
                });
                let dayData = Object.assign({}, self.dayObject);
                dayData.date = tempDate.toISODate();
                dayData.value = d.doc_count;
                dayData.forecast = false;
                sortedData.push(dayData)
            }
        });
        // Parse forecasted data to dayData objects and add to sortedData
        const firstDt = DateTime.local();
        forecastData.forEach((d, index) => {
            if (d) {
                let dayData = Object.assign({}, self.dayObject);
                dayData.date = firstDt.plus({ days: index }).toISODate();
                dayData.value = d;
                dayData.forecast = true;
                sortedData.push(dayData);
            }
        });
        return sortedData;
    }

```