CHALMERS

Optimera användningen av virtuella maskiner i Azure med maskininlärning
Optimize the utilization of virtual machines in Azure with machine learning

Examensarbete inom Data- och Informationsteknik

KEVIN HEDBERG GRIFFITH
ERIK NGUYEN

Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2016

EXAMENSARBETE

Optimera användningen av virtuella maskiner i Azure med maskininlärning

Kevin Hedberg Griffith
Erik Nguyen

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA

Göteborg 2016

Optimera användningen av virtuella maskiner i Azure med maskininlärning
Kevin Hedberg Griffith
Erik Nguyen

Examinator: Christer Carlsson

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

Institutionen för Data- och Informationsteknik
Göteborg 2016

FÖRORD

Vi är glada över möjligheten att ha fått utföra vårt examensarbete på Atea Global Services. Vi har dem att tacka för de framsteg och kunskaper som vi har uppnått under denna period. Examensarbetet har varit lärorikt och gett oss kunskaper i både Microsoft Azure och maskininlärning. Vi vill tacka för framtagandet av ett intressant examensarbete samt för gästvänligheten och hjälpen från handledarna Carl Stein, Andreas Fransson och Andreas Warg samt deras arbetskollegor.

Vi vill också tacka vår handledare på skolan Emil Axelsson som har varit till stor hjälp gällande rapportskrivandet under examensarbetets gång.

Detta examensarbete utgör 15 högskolepoäng och har utförts som ett sista delmoment på det 3-åriga dataingenjörsprogrammet som omfattar 180 högskolepoäng vid Chalmers Tekniska Högskola.

SAMMANFATTNING

Denna rapport beskriver ett arbete som undersöker möjligheten att optimera användningen av virtuella maskiner i Microsoft Azure med hjälp av maskininlärning. Arbetet har gjorts på företaget Atea Global Services (AGS). Virtuella maskiner är en Azure tjänst som AGS använder. Dock kan dessa virtuella maskiner vara i drift utan att verkligen användas. Tjänster i Azure är inte är gratis och vid användning av t.ex. virtuella maskiner debiteras företag per minut. Detta betyder att AGS betalar onödiga kostnader för de virtuella maskiner som är i drift och inte används. Med hjälp av en Azure-tjänst vid namn Machine Learning Studio kunde ett användarmönster för när en virtuell maskin användes tas fram. En applikation har skapats som stänger av eller slår på en virtuell maskin baserat på användarmönster från Machine Learning Studio. AGS kan välja att fortsätta utveckla arbetet eller ta nytta av det för att minska kostnaderna för användandet av deras virtuella maskiner.

SUMMARY

This report describes a project that is about examining the possibilities to optimize the utilization of virtual machines in Microsoft Azure using machine learning. The thesis has been done at the company Atea Global Services (AGS). Virtual machines is a Azure service that AGS uses. However could these virtual machines be running without really being used. Services in Azure are not for free and when using a service like virtual machines companies are being charged be per minute. This means that AGS pays unnecessary expenses for the virtual machines that are running when they are not being used. Using an Azure service called Machine Learning Studio, a user pattern for when a virtual machine was being used was developed. An application has been developed that turns on or off a virtual machine based on user patterns from Machine Learning Studio. AGS can choose whether they want to continue working on the project or to take advantage of it right away to cut back on costs.

INNEHÅLLSFÖRTECKNING

BETECKNINGAR

ML Studio - Azure Machine Learning Studio
AGS - Atea Global Services
CPU - Central Processing Unit
JSON - JavaScript Object Notation

1 INLEDNING

1.1 Bakgrund

Molntjänster ger möjligheten till att hantera datalagring, program och processorkraft på en extern resurs som tillhandahålls av leverantörer i form av tjänster över Internet.

Användningen av molntjänster har dragit till sig stor uppmärksamhet och har blivit alltmer populära och idag erbjuds det tjänster både till privatpersoner och företag. Speciellt har en ökning i nyttjande av molntjänster skett hos företag. Fördelen är att företagen inte behöver investera och underhålla egen IT-utrustning. Istället kan företagen betala för att lägga ut delar av sin IT-verksamhet på molnet.

Atea Global Services (AGS) inriktar sig på IT-infrastruktur och arbetar med att leverera tjänster som klienthantering och applikationspaketering till kunder över hela världen. Som många andra företag använder AGS molnet för delar av sin verksamhet och har valt att samarbeta med Microsoft och dess service Azure. Azure är en service som erbjuder molntjänster som t.ex. analysering, databearbetning, databaser, mobila applikationer och lagring.

All personal på AGS använder sig av virtuella maskiner från Azure. Virtuella maskiner kan liknas vid en fysisk dator med skillnaden att virtuella maskiner är i molnet och kräver Internetuppkoppling och inloggningsuppgifter för att upprätta en anslutning.

De tjänster som Azure erbjuder är inte gratis och vid användning av t.ex. virtuella maskiner debiteras företag per minut. De virtuella maskiner som AGS använder stängs aldrig av. Detta betyder att AGS betalar onödiga kostnader för de virtuella maskiner som är i drift men inte används.

1.2 Syfte

AGS ser ett problem i att betala onödiga kostnader för virtuella maskiner när de inte används. Syftet med detta arbete är att lösa detta problem genom att minska kostnaderna för de virtuella maskinerna i Azure som AGS använder.

1.3 Mål

Azure erbjuder en tjänst vid namn Machine Learning Studio (ML Studio) som utnyttjar maskininlärningsalgoritmer (algoritmer) för mönsterigenkänning. ML Studio skall användas för att ta fram ett användarmönster för hur en virtuell maskin används och på sådant sätt kunna förutse tider för påslagning och avstängning. En applikation ska skapas som kommunicerar med ML Studio samt hanterar virtuella maskiner. Målsättningen med arbetet är att effektivisera användandet av virtuella maskiner i Azure.

1.4 Avgränsningar

- Arbetet kommer att ske i Azure och Visual Studio.
- Maskininlärning kommer vara begränsad till ML Studio.
- Arbetet är begränsat till följande språk: Python, R, PowerShell och C# med .NET-ramverket.

1.5 Frågeställningar

- Vilka lämpliga metoder finns det för att undersöka användarmönstret för en virtuell maskin?
- Vilken användardata skall övervakas för att avgöra om en virtuell maskin används?
- På vilket sätt skall användardatan hanteras för projektets ändamål?
- Hur skall ML Studio användas för att effektivisera användningen av en virtuell maskin?
- Hur sker avstängning och påslagning av virtuella maskiner efter att ha fått resultat från ML Studio?

1.6 Disposition

- I kapitel 2 förklaras hur arbetet är upplagt och beskriver kortfattat de olika delarna som arbetet är uppbyggt på.

- Kapitel 3 ger en teknisk förklaring till de delar som står grund till arbetet.

- Kapitel 4 går igenom hur vi gått tillväga för att skapa de olika delarna och upprätta kommunikation sinsemellan för att slutligen komma fram till resultatet.

- I kapitel 5 presenteras resultatet som har åstadkommits. Bland annat besvaras frågorna i frågeställningen.

- I kapitel 6 diskuteras de problem som uppstått under arbetets gång samt förbättringar och möjlig vidareutveckling.

2 METOD

Huvudfokus kommer att vara på maskininlärning och ML Studio. Följande programmeringsspråk kan komma att användas: Python, R, PowerShell och C#. Applikationen kommer att skrivas i programutvecklingsmiljön Microsoft Visual Studio i programmeringsspråket C# med .NET-ramverket.

Utöver en applikation, ML Studio och virtuella maskiner skall följande verktyg användas: databas, SQL server och runbooks. Arbetet är uppdelat i fem delar varav alla delar utom applikationen är tjänster i Azure.

2.1 Användarmönster

Eftersom uppgiften för applikationen är att stänga av och slå på en virtuell maskin måste ett mönster för hur den används tas fram. Det saknas exakta tillvägagångssätt för att undersöka hur en virtuell maskin används, därför kommer mycket fokus att ligga på att lösa denna uppgift.

2.2 Datahantering

För att spara och hantera användardata från de virtuella maskinerna behövs en databas. Vidare behöver denna data hanteras för att förbereda en tabell som skall användas i ML Studio.

2.3 Azure Machine Learning Studio

Ett experiment i ML Studio bygger på att en algoritm körs på ett dataset vilket resulterar i en modell. I detta arbetet används ML Studio för att hitta ett användarmönster och förutse tider för avstängning och påslagning av virtuella maskiner. När ett experiment körts och en modell tagits fram skall modellen testas för att sedan distribueras och användas av en applikation.

2.4 Runbooks

Automation är en tjänst i Azure som gör det möjligt att automatisera övriga tjänster i Azure. Med Automation kan så kallade runbooks skapas. En runbook (även kallad PowerShell-skript) gör det möjligt att skapa, distribuera, övervaka och underhålla tjänster i Azure genom att utföra Azure-kommandon.

2.5 Arbetsmetoder

De olika delarna som används i arbetet kan liknas vid byggstenar. Innan samtliga byggstenar kan implementeras behöver ytterligare tid tillbringas för att lära sig om hur dessa fungerar. Tidigare erfarenheter saknas i de olika delarna vilket gör det svårt att uppskatta hur mycket tid samtliga delar behöver för att implementeras och om dessa är det optimala tillvägagångssättet. Detta medför att arbetsmetoder som t.ex. Scrum inte används. Istället hålls möten veckovis för att diskutera om vad som har åstadkommits och vad som behöver förbättras.

Mycket tid kommer gå åt till ML Studio och maskininlärning och därför prioriteras dessa delar högst. Övriga byggstenar är svåra att tidsuppskatta.

3 TEKNISK BAKGRUND

3.1 Maskininlärning

Maskininlärning är ett område inom datavetenskapen, som utvecklats från studier av mönsterigenkänning, där konstruktioner av algoritmer som kan göra förutsägelser på data undersöks [1]. Datamängder undersöks för att hitta mönster, därefter genereras en kod som känner igen dessa mönster i ny data [2]. Den genererade koden (modell) kan sedan användas av applikationer för att göra förutsägelser.

Användningsområdena för maskininlärning är oändliga [2]. Maskininlärning kan användas för spårning av bedrägeri, förutse ett företags framtida intäkter, ta beslut om när en jetmotor behöver underhållas eller, som i detta arbete, för att förutse tider för påslagning och avstängning av virtuella maskiner. Bild 3.1 sammanfattar maskininlärnings processen från rådata till en modell som utgår från dessa data.

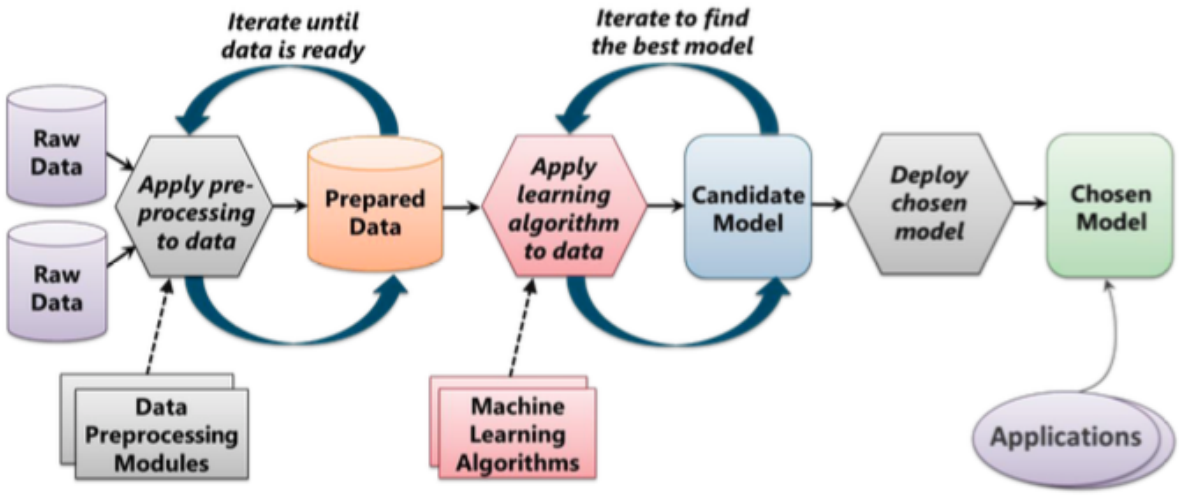


Bild 3.1. Maskininlärningsprocessen - från rådata till modell [2].

Processen börjar med rådata och mer data leder till bättre resultat. Datan är i formen av ett dataset (tabell). Det är viktigt att välja rätt data och att urskilja vad som är nödvändigt och relevant för ändamålet. Rådatan måste förbehandlas eftersom dubbletter kan finnas eller kolumner kan sakna data [2]. Nästa steg undersöker vilken algoritm som tillsammans med datan ger bäst resultat. Efter undersökningar tas det bästa resultatet fram i form av en modell. Sista steget distribuerar modellen vilket gör det möjligt för applikationer att använda den tränade algoritmen som modellen implementerar.

3.2 ML Studio

ML Studio är en tjänst i Azure som utnyttjar maskininlärning och är ett grafiskt verktyg som gör det möjligt att skapa experiment. Experimenten använder en drag-och-släpp metod på en arbetsyta och underlättar hantering av maskininlärningsprocessen. Bild 3.2 visar hur ML Studio använder maskininlärningsprocessen (se bild 3.1).

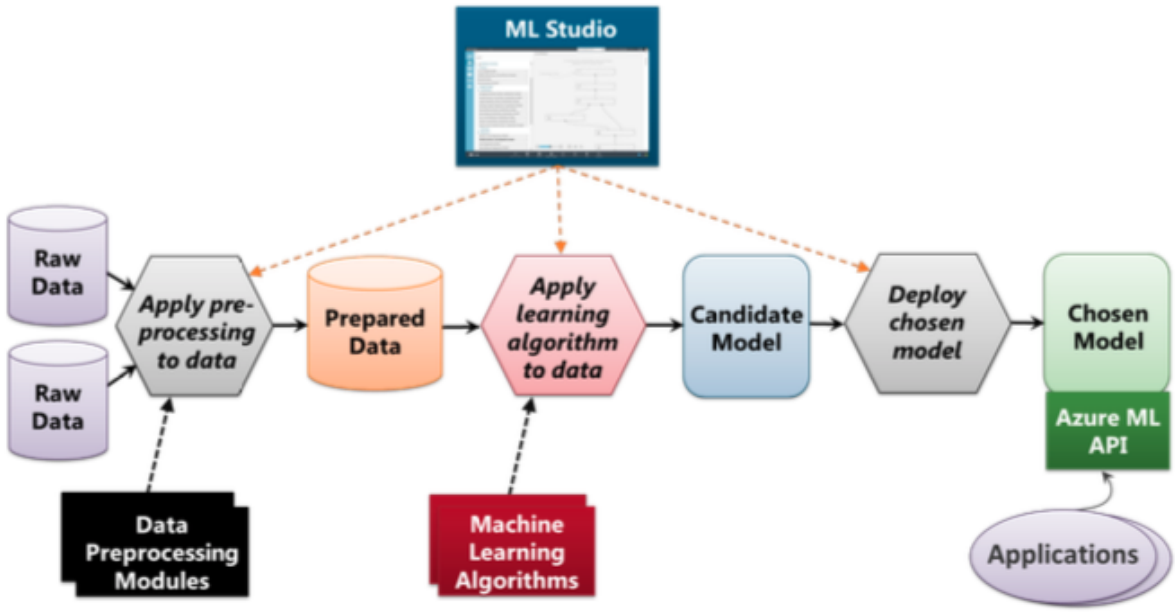


Bild 3.2. Visar hur ML Studio använder delar av maskininlärningsprocessen [2].

ML Studio tillhandahåller moduler för algoritmer, förbehandling av data samt distribuering av modeller för användning i applikationer.

Bild 3.3 visar ett exempel på ett experiment i ML Studio där moduler hämtas från menyn till vänster. Exempel på moduler för förbehandling är borttagning av dubbletter och borttagning av rader som saknar värden (bild 3.4).
Algoritmerna är uppdelade i kategorierna Classification, Regression, Cluster och Anomaly Detection. Dessa används för att lösa olika typer av problem:

Classification - Väljer utifrån indata exakt vilken av två eller fler fördefinierade kategorier som denna indata faller in på [2].

Regression - Genom att undersöka förhållanden mellan tidigare efterfrågningar (målvärde) och övrig indata (funktioner) kan regression algoritmer användas för att förutse framtida efterfrågningar [2].

Cluster - Undersöker tidigare data, hittar betydelsefulla klungor och skapar en modell som listar ut till vilken klunga som ny data skall tilldelas [2].

Anomaly Detection - Upptäcker avvikande mönster i data som inte stämmer överens med det förväntade beteendet [3].

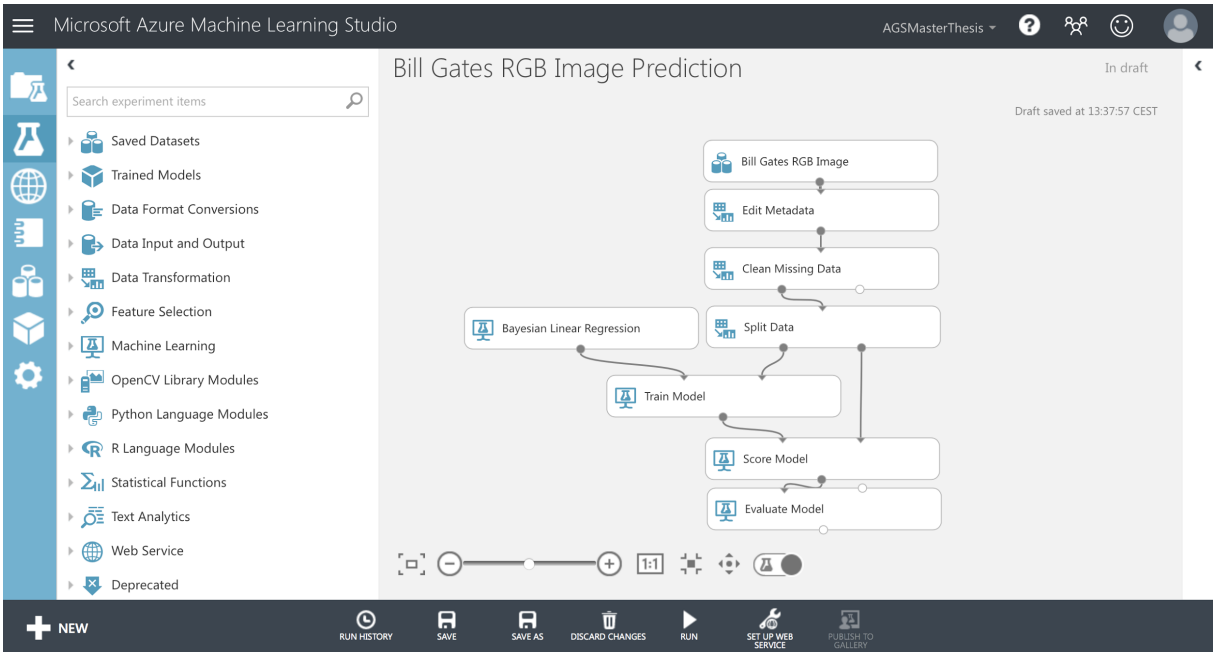


Bild 3.3. Exempel på experiment i ML Studio, modulerna hämtas i menyn till vänster.

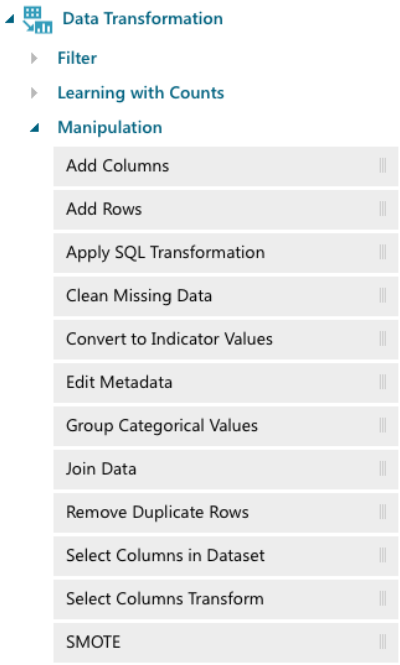


Bild 3.4. Förbehandlingsmoduler i ML Studio.

Modulen Train Model (bild 3.3) kräver att en kolumn anges (målkolumnen). Målkolumnen som anges är den som modellen skall hitta förutsägelser på. Övriga kolumner (funktioner) används för att träna vald algoritm och skapa en modell (bild 3.5) [2].

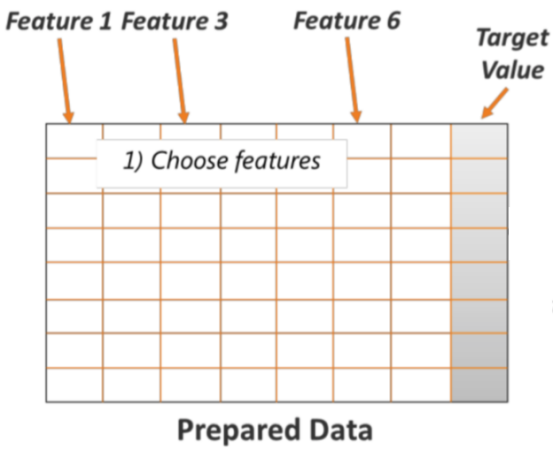


Bild 3.5. Vid träning av algoritmer måste de kolumner som skall användas i träningen anges samt vilken kolumn det är som skall förutses [2].

Vid träning av en algoritm på data måste datan delas upp (Split Data modul i bild 3.3). En del används för att träna algoritmen (träningsdata) och den andra används för testning (testdata) (se bild 3.6). Vanlig uppdelning av data är 70-75% träningsdata och 30-25% testdata. Detta förser algoritmen med tillräckligt mycket träningsdata samt ger tillräckligt mycket testdata för att bedöma hur bra modellen är. När en modell tagits fram kan den testas genom att testdatan jämförs med den faktiska datan (bild 3.7) [2].

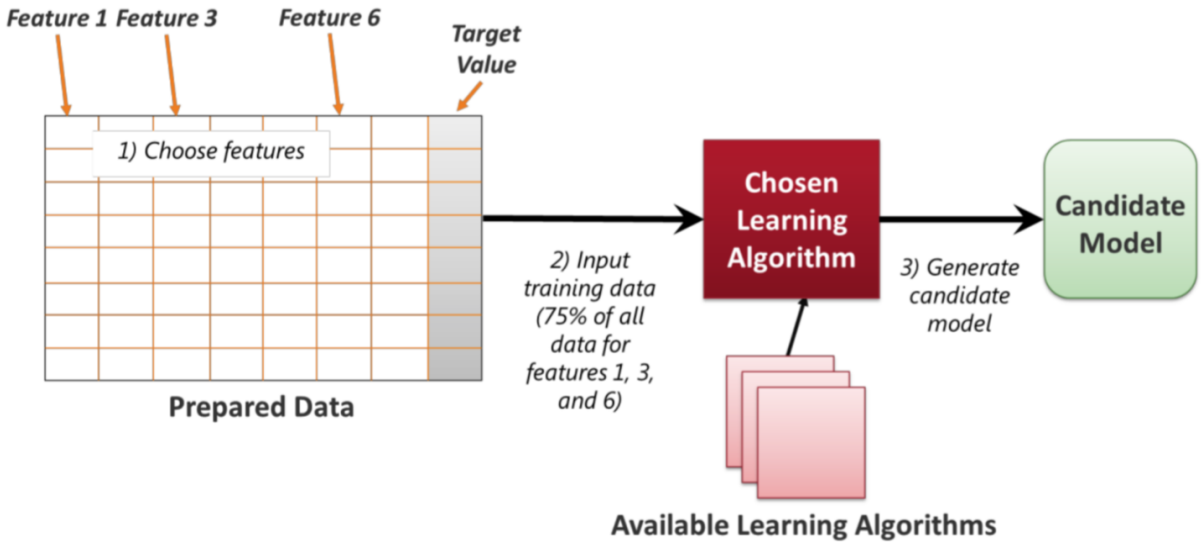


Bild 3.6. När målkolumn valts delas datan upp i 2 delar, träningsdata och testdata. Därefter tränas en algoritm och en prediktiv modell skapas.

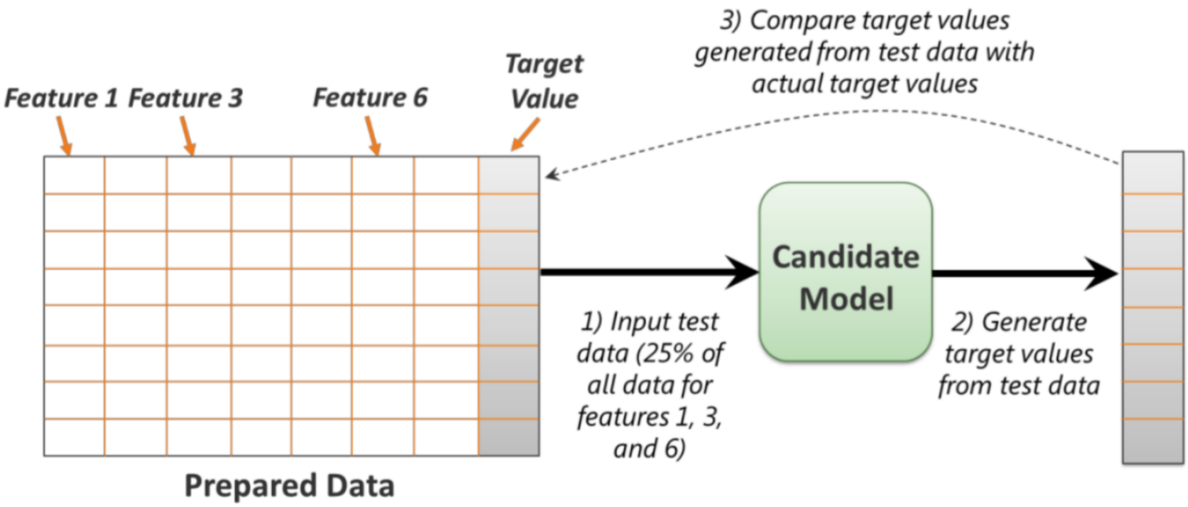


Bild 3.7. När en modell skapats kan den testas med hjälp av testdatan för att jämföras med den aktuella målkolumnen [2].

När en slutgiltig modell tagits fram är sista steget att distribuera den för att göras användbar i applikationer. Med två moduler (bild 3.8) kan en modell i ML Studio distribueras. I ML Studio kallas det uppsättning av webb-service (bild 3.9). En webb-service tilldelas en unik API-nyckel som används i applikationer. Applikationen kan skicka förfrågan med samma inparametrar som algoritmen tränats på och modellen förutser ett svar som skickas tillbaka.

Bild 3.8. Webb-service moduler som möjliggör för modell i ML Studio att användas av applikationer.

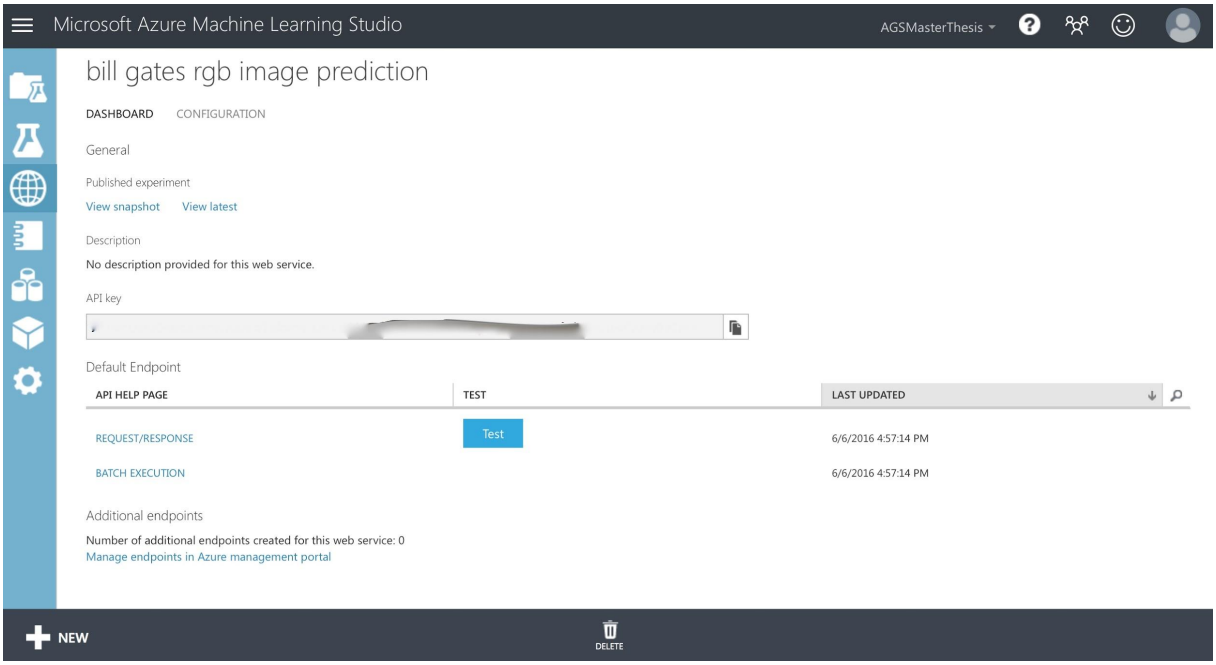


Bild 3.9. Exempel på webb-service startsida för en modell vid namn "bill gates rgb image prediction".

3.3 Virtuella maskiner & varningsregler

Virtuella maskiner har samma funktionalitet som en fysisk dator med skillnaden att en virtuell maskin kan köras på vilken dator som helst så länge det finns en Internetuppkoppling och inloggningsuppgifter. De virtuella maskiner som kommer att användas är i Azure. Varje virtuell maskin kan monitorera[1] mätvärden som t.ex. CPU-användning, skickade eller mottagna TCP-segment, minnesanvändning och antal webb-anslutningsförsök. Bild 3.10 visar ett exempel på ett diagram över CPU-användningen.

[1] Monitorera - övervaka en maskins eller patients tillstånd med stöd av mätvärden. Monitor - apparat som övervakar en process eller dylikt och varnar vid avvikelser från det normala.

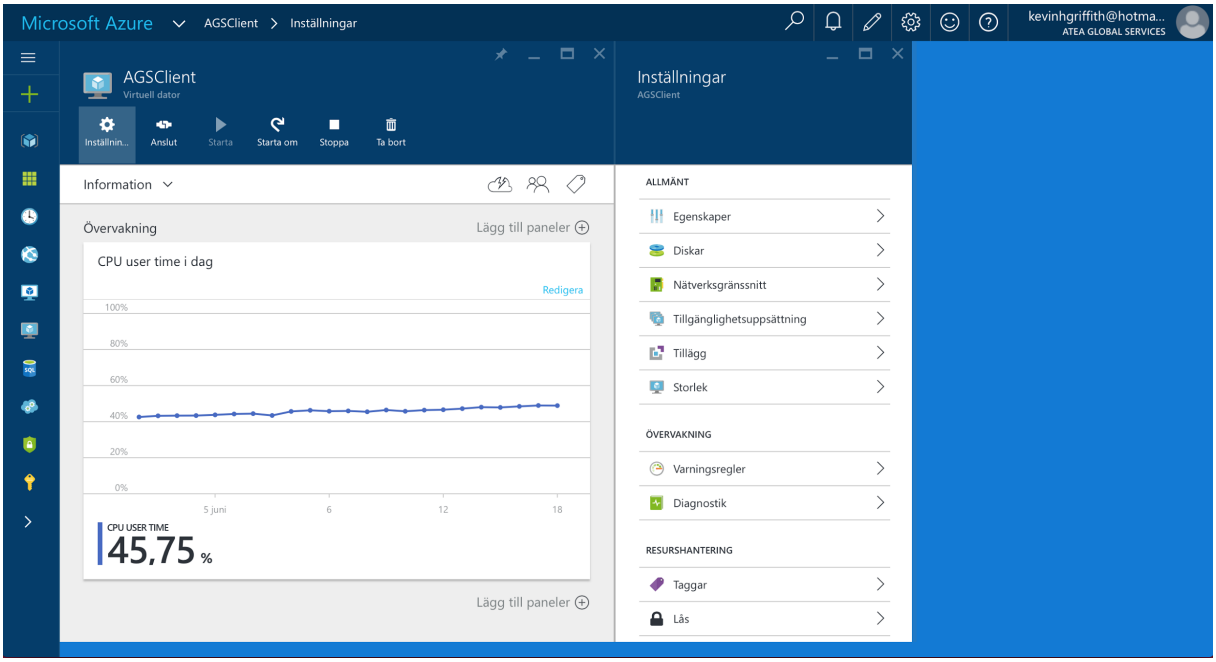


Bild 3.10. Startsida & inställningar för en virtuell maskin samt diagram som visar CPU-användningen.

För att undersöka hur en virtuell maskin används kan så kallade varningsregler sättas upp som monitorerar över dessa. När en varningsregel triggas skickas ett meddelande i form av ett JSON-objekt [4] till en mottagare. Exempel på varningsregel kan vara att monitorera CPU-användningen och undersöka ifall en virtuell maskin stiger över 10% under ett visst tidsintervall. Om så är fallet triggas varningsregeln och sänder ut ett JSON-objekt med information som kan tas hand om på olika sätt.

Bild 3.11 visar startsidan för varningsregler, dess inställningar, vilka varningsregler som finns tillgängliga samt när varningsreglerna senast har blivit triggade (kolumnen SENAST AKTIV) i Azure. Orange symbol i fliken Varningsregler indikerar att en varningsregel har triggats och grön symbol motsatsen.

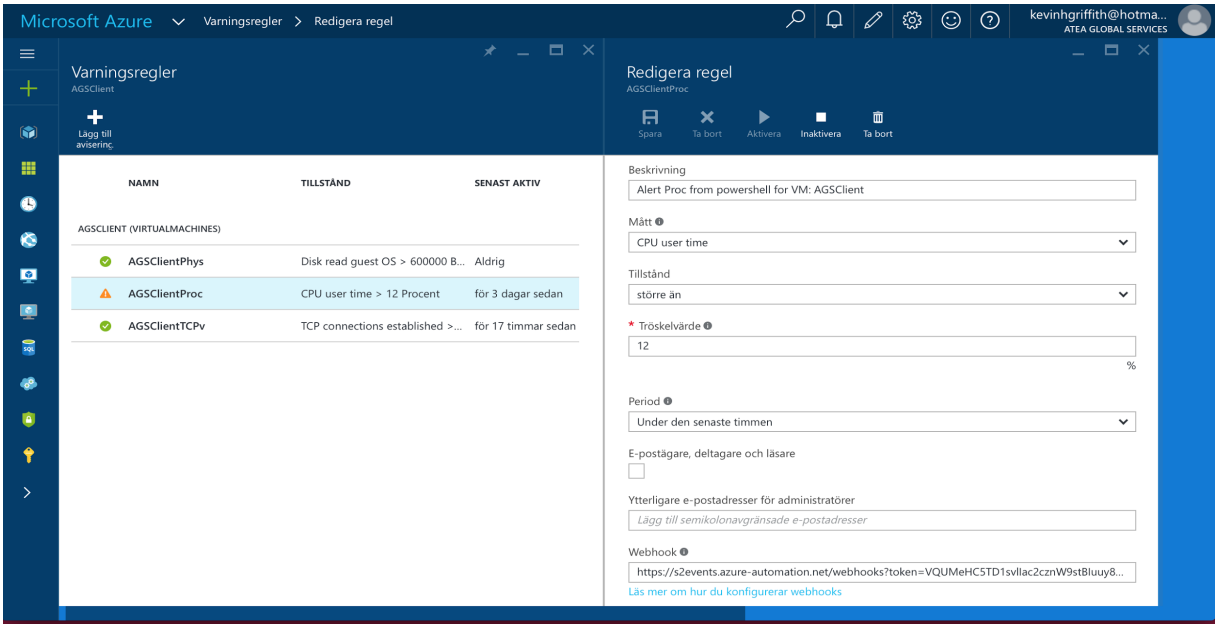


Bild 3.11. Startsida, inställningar och status för varningsregler.

3.4 Runbooks

Automation är en tjänst i Azure som gör det möjligt att hantera och styra övriga tjänster i Azure med hjälp av runbooks [5]. Operationer som kan utföras är t.ex. start och stopp av virtuella maskiner. Runbooks kan köras manuellt via Azure, genom att schemalägga en tid för körning eller med HTTP-begäran [6]. Bild 3.12. visar startsidan för Azure Automation samt en lista på skapade runbooks.

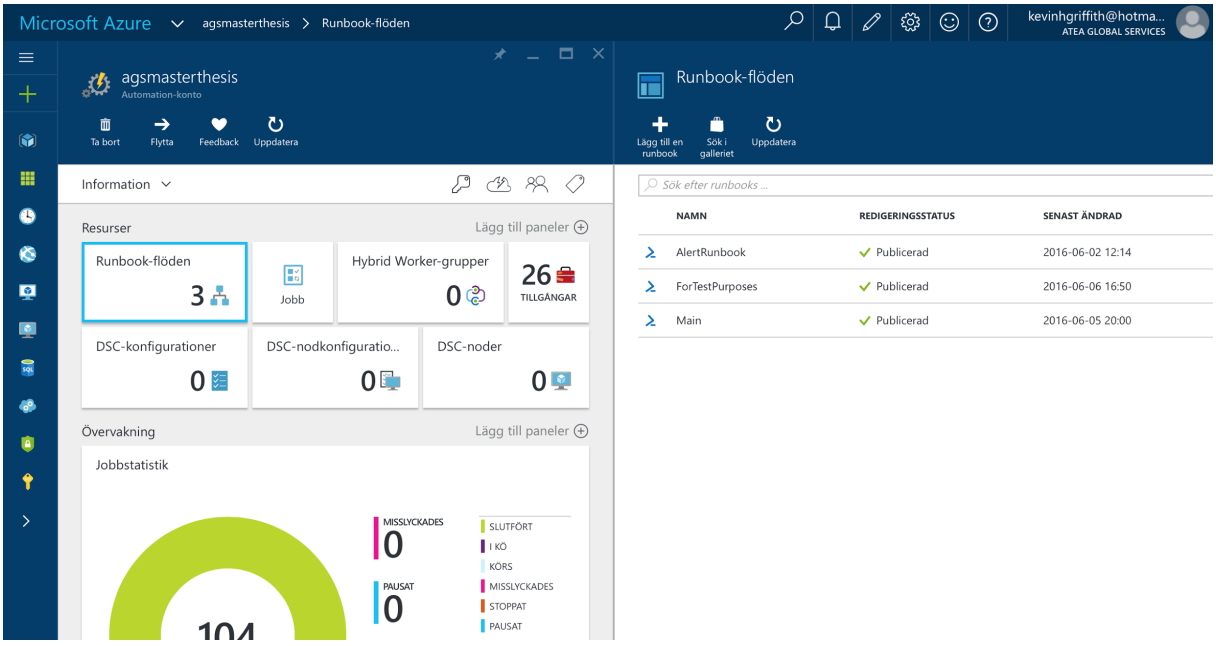


Bild 3.12. Startsida för Automation och lista på skapade runbooks.

3.5 Webhooks

Webhooks är ett tillvägagångssätt för en applikation att förse andra applikationer med information i realtid [7]. Det är inte en tjänst i Azure, dock används det av tjänster i Azure. Webhooks triggas när händelser sker, t.ex. när kod visas upp i ett repository på Github [8] eller när ett nytt mail läggs i inkorgen. Innan en webhook kan användas måste den förses med en unik URL som används för att skicka HTTP-begäranden till [6].

I detta arbetet används webhooks som stöd för att vid specifika händelser starta runbooks. En webhook på en runbook fungerar som en brevlåda mottaglig för data-objekt (t.ex. JSON-objekt). Ett JSON-objekt skickas som en HTTP-begäran för att starta den runbook som har den uppsatta webhooken. Bild 3.13 visar uppsättning av webhook på en runbook vid namn ForTestPurposes. Notera att Webhook är markerad i fliken till vänster.

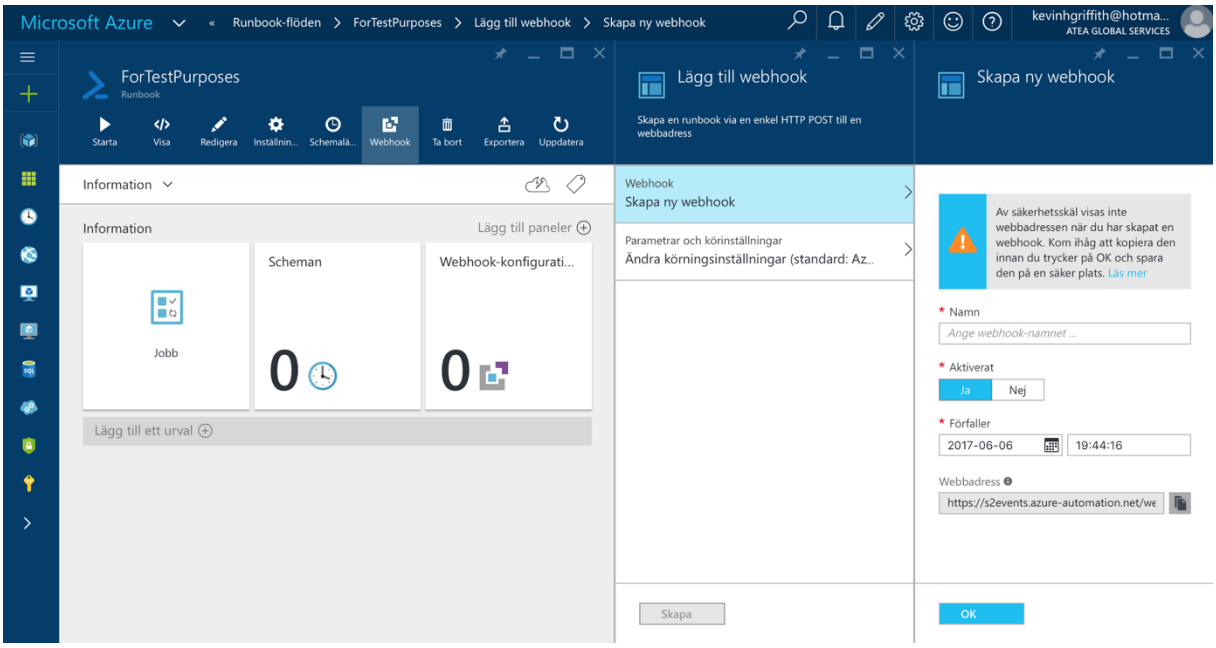


Bild 3.13. Vänster flik visar startsidan för runbooks, de två högra flikarna visar uppsättning av webhook på en runbook.

3.6 Visual Studio

Visual Studio är en programutvecklingsmiljö från Microsoft [9]. Visual Studio kan användas för att skapa allt från datorprogram till applikationer för mobila enheter såsom iOS, Android och Windows [10]. Det huvudsakliga programmeringsspråket är C# med .NET-ramverk. Utöver detta kan Visual Studio användas för att ansluta till och hantera databaser.

4 GENOMFÖRANDE

En systemarkitektur ritades upp (kallad Näringskedjan, bild 4.1) på den primära versionen av arbetet. Näringskedjan visar de olika byggstenarna som arbetet är uppbyggt på och kommunikationen sinsemellan. I detta kapitel kommer Näringskedjans olika delar att gås igenom och ändringar kommer att ske där också kommunikationsmodulen kommer att identifieras. Ändringarna visas med nya bilder och i kapitel 5 (resultat) visas den slutgiltiga versionen av Näringskedjan.

Näringskedjan tillsammans med tidsplanen (se appendix) är den arbetsmetod som skall hållas genom arbetets gång.

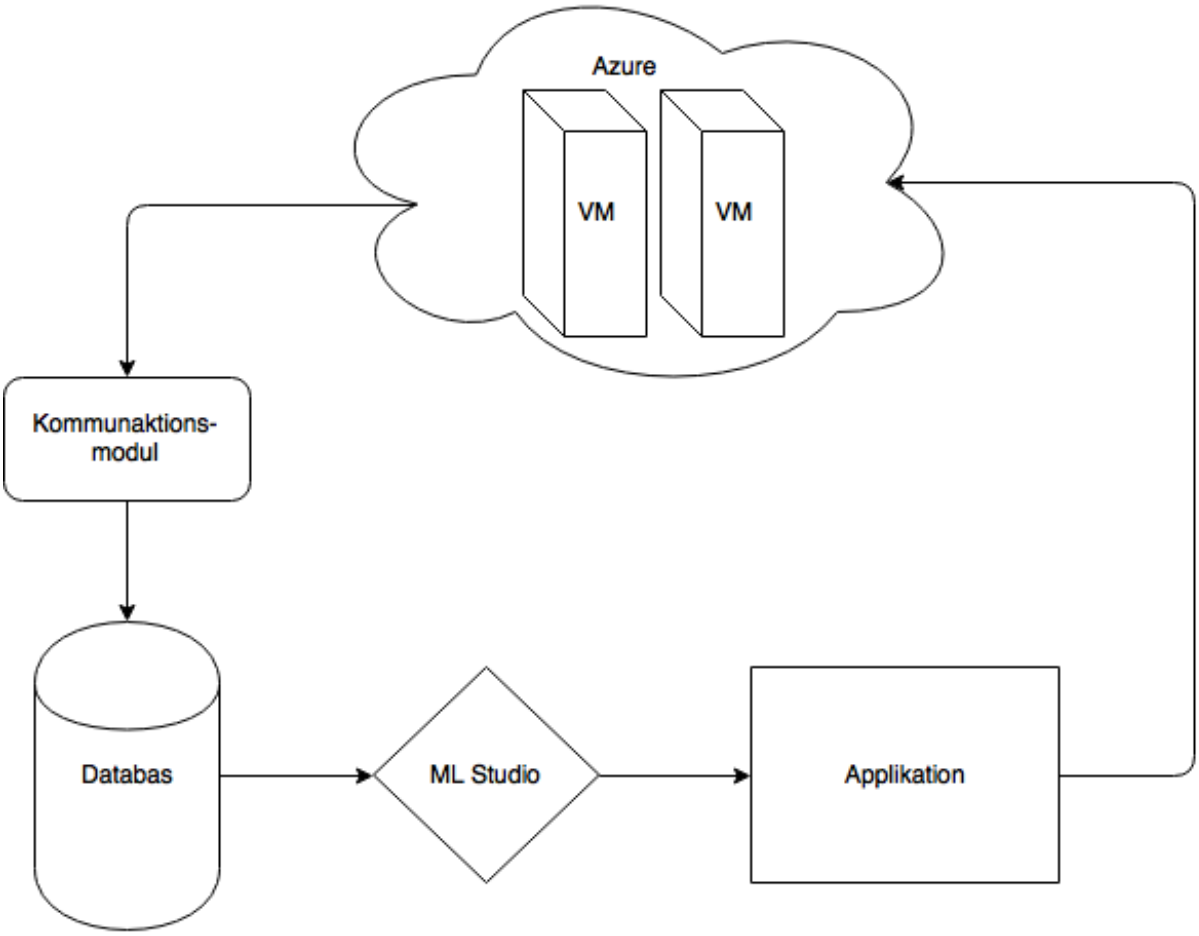


Bild 4.1. Primär version av Näringskedjan som representerar byggstenarna i arbetet.

4.1 Analysering av användningsmönster

För att hitta ett användningsmönster har undersökningar gjorts för att få svar på vilka mätdata som definierar när en virtuell maskin används. När mätdata hittats har ytterligare undersökningar gjorts för att ta reda på vilka data som faktiskt kan hämtas från virtuella maskiner i Azure.

4.1.1 Lämplig mätdata från en virtuell maskin i Azure

Ett problem med virtuella maskiner är att de inte beter sig på samma sätt som en dator. När en fysisk dator är i drift och inte kör något program ligger CPU-användandet på en låg och stabil nivå (utan större förändringar). Detta medför att det är enkelt att avgöra ifall någon använder datorn genom att enbart avläsa CPU-användandet. Nivån för CPU-användningen i virtuella maskiner har noterats öka till nivåer i höjd med en användarnivå trots att den inte används i drift. Det kan bero på automatiska uppdateringar i den virtuella maskinen eller oväntade störningar på Microsofts servrar. Detta beteende gör det svårt att enbart kunna förlita sig på CPU-användningen för att avgöra ifall någon använder en virtuell maskin och därför måste fler mätvärden tas med.

CPU-användning, läsning och skrivning av data på disk, minnesanvändning och skickade och mottagna TCP-segment är mätdata som behövs för att särskilja mellan att en virtuell maskin i används i drift eller inte [11].

En ökning på CPU-användandet där varken TCP- eller diskanvändningen påverkas kan betyda att den virtuella maskinen inte används i drift. Förändringar i TCP-användningen sker endast vid uppkoppling mot Internet och inte när offline-arbeten utförs. Förändringar i diskanvändningen sker t.ex. när filer öppnas eller program körs och CPU-användningen påverkas när både enkla och mer krävande arbeten utförs (t.ex. inloggning till virtuell maskin).

Godtyckliga resultat kunde avläsas trots monitorering av endast CPU-användningen. Ett tillvägagångssätt med enbart monitorering över CPU-användning valdes då det inte var självklart hur alla mätvärden skulle kombineras för att tillsammans hanteras vidare i databasen och ta fram bra starttider och sluttider. En kombination av dessa mätdata ger definitivt ett bra omdöme för om en virtuell maskin används. Därför beslutades det att implementera denna kombination när fler delar av arbetet gjorts.

4.1.2 Varningsregler för en virtuell maskin

Ett tillvägagångssätt för att bli informerad när CPU-användandet ändras behövs för utifrån det kunna agera. Av alla funktioner i Azure för en virtuell maskin och dess mätdata hittades en lösning vid namn varningsregler. Varningsregler fungerar som en övervakning på mätdata (monitorering). Varningsreglerna sätts upp med en tröskel. Om tröskeln passeras triggas varningsregeln och ett meddelande skickas via email eller webhooks med information om t.ex. vilken mätdata och vilken virtuell maskin som har triggat varningsregeln samt om tröskeln passerats eller lösts.

Ytterligare undersökningar gjordes på CPU-användningen för att bestämma hur hög tröskeln skall vara. Om tröskeln för en varningsregel är 10% och CPU-användningen går över 10% under en viss tidsperiod skall en varningsregel triggas. Större än eller lika med 10% betyder att en virtuell maskin används och mindre än 10% motsatsen. Denna tröskeln såg till att inga mindre plötsliga förändringar i CPU-nivån skulle trigga en varningsregel.

Utskick av email användes i början på grund av saknad kunskap om webhooks. Men för att utnyttja informationen från varningsmeddelandena blev webhooks det självklara och slutgiltiga valet. Hur informationen hanteras vidare beskrivs i nästa avsnitt.

4.2 Datahantering

En databas och SQL-server skapas i Azure då databasen smidigt kan anslutas till Visual Studio och hanteras på samma ställe som applikationen.

4.2.1 Webhooks

Anslutningen mellan virtuella maskiner och databasen görs med hjälp av webhooks. En kommunikationsmodul som kan ta emot och hantera informationen från varningsregler på virtuella maskiner behövs för att sedan lägga in det i databasen.

En första implementation för att hantera webhooks blev Zapier [12]. Zapier är en webbapplikation som gör det möjligt att ansluta applikationer med varandra och automatisera uppgifter genom webhooks. Efter uppsättning av Zapier fanns det en automatiserad anslutning mellan varningsregler och databasen (bild 4.2). Processen från att en varningsregel triggas till att informationen läggs in som en ny rad i databasen kan nu ske automatiskt.

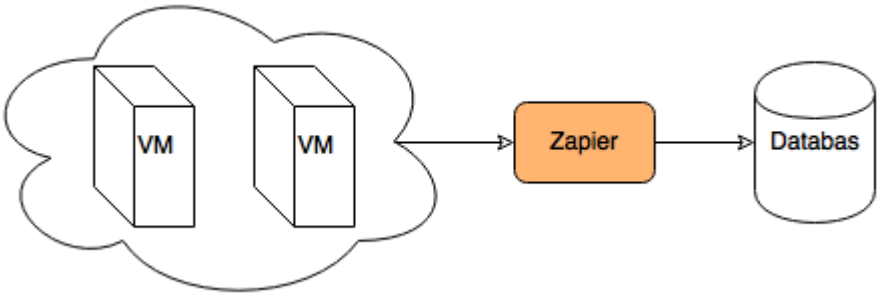


Bild 4.2. Upprättad anslutning mellan varningsregler på virtuella maskiner och databas med hjälp av Zapier.

4.2.2 Runbook för varningsregler

Den slutgiltiga lösningen blev Azure-tjänster Automation och dess runbooks. Detta för att Zapier inte är gratis. Det var tänkt att skapa runbooks för varje virtuell maskin men detta var inte en optimal lösning då möjligheten till att på ett smidigt sätt lägga till och monitorera över nya virtuella maskiner försvårades. Istället togs en lösning fram där endast en runbook skulle utgöra samma funktionalitet som Zapier. Fördelarna är att en Azure tjänst används samt att tjänsten är gratis. Runbooken har uppgiften att agera som mottagare för varningsregler, ta hand om informationen och extrahera användbar information för att sedan sätta in informationen i databasen (bild 4.4).

```powershell
param (
    [object]$WebhookData
)
# If runbook was called from Webhook, WebhookData will not be null.
if ($WebhookData -ne $null) {
    # Collect RequestBody of WebhookData
    $WebhookBody    =   $WebhookData.RequestBody

    #Collect SQL information
    $Server = Get-AutomationVariable -Name 'SqlServer'
    $Port = Get-AutomationVariable -Name 'SqlServerPort'
    $Database = Get-AutomationVariable -Name 'Database'
    $Table = Get-AutomationVariable -Name 'Table'

    # Obtain WebhookBody conatining the AlertContext
    $WebhookBody = ConvertFrom-Json -InputObject $WebhookBody

    # Obtain vm, status and metricName
    $VM = [object]$WebhookBody.context.resourceName
    $Status = [object]$WebhookBody.status
    $metricName = [object]$WebhookBody.context.condition.metricName

    # Get the username and password from the SQL Credential
    $SqlUsername = (Get-AutomationPSCredential -Name 'SqlCredential').GetNetworkCredential().username
    $SqlPass = (Get-AutomationPSCredential -Name 'SqlCredential').GetNetworkCredential().password

    # Create connection to Master DB
    $MasterDatabaseConnection = New-Object System.Data.SqlClient.SqlConnection
    $MasterDatabaseConnection.ConnectionString = "Data Source=$Server;Initial Catalog=$Database;
     + "Integrated Security=False;User ID=$SqlUsername;Password=$SqlPass;Connect Timeout=60;
     + "Encrypt=False;TrustServerCertificate=False"
    $MasterDatabaseConnection.Open()

    # Create command
    $MasterDatabaseCommand = New-Object System.Data.SqlClient.SqlCommand
    $MasterDatabaseCommand.Connection = $MasterDatabaseConnection
    #The value 1 is a default value that will be changed by the trigger in $Table
    $MasterDatabaseCommand.CommandText =
    "INSERT INTO dbo.$Table(id, vm, status, metricType) VALUES (1, '$VM', '$Status', '$metricName')"

    # Execute the query
    $MasterDatabaseCommand.ExecuteNonQuery()

    # Close connection to Master DB
    $MasterDatabaseConnection.Close()
}
```

Bild 4.3. Runbook som tar emot data från varningsregler, extraherar ut användbar information och utför en insättning till en databas.

Rådatan som kommer direkt från varningsregeln skall först filtreras eftersom endast en liten del av information är användbar för ML Studio. Den datan som är användbar och som extraheras är identifierare för vilken virtuell maskin som har triggat varningsregeln (vm), status på varningsregeln (antingen Activated eller Resolved) (status) och identifierare för vilket mätvärde som har triggats (metricType). Databasen kommer i sin tur lägga till egna kolumner innan en insättning till en tabell i databasen sker och detta förklaras mer i kommande avsnitt. Bild 4.4 visar anslutningen mellan varningsregler och databas då Zapier ersätts.

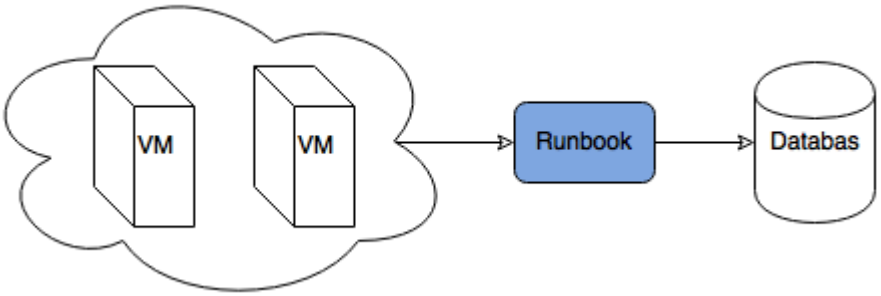


Bild 4.4. Upprättad anslutning mellan varningsregler på virtuella maskiner och databas med hjälp av runbooks i Azure.

4.2.3 Utformning av databas

Databasen är en central del i arbetet då användardata från virtuella maskiner sparas här och användas i ML Studio. Data från varningsreglerna sparas i en tabell vid namn AlertsTable. Dock är det inte denna tabell som ML Studio hämtar data från. För att strukturera upp databasen samt forma en tabell anpassad att användas i ML Studio behöver triggers och ytterligare tabeller skapas.

Slutgiltiga versionen av databasen visas i bild 4.5. Tabellernas namn och funktion är följande:

AlertsTable - Den primära tabellen som tar emot alla varningsregler (aktiva och lösta).

CPUTable - Sparar alla varningsregler med CPU-användning som mätvärde.

TCPTable - Sparar alla varningsregler med TCP-användning som mätvärde.

DiskReadTable - Sparar alla varningsregler med disk-användning som mätvärde.

FinalTable - Sparar data som skall användas i ML Studio. Här kombineras CPUTable, TCPTable och diskReadTable. En rad i denna tabell ger information om start-och sluttid för en specifik virtuell maskin för ett visst datum och dagnummer.

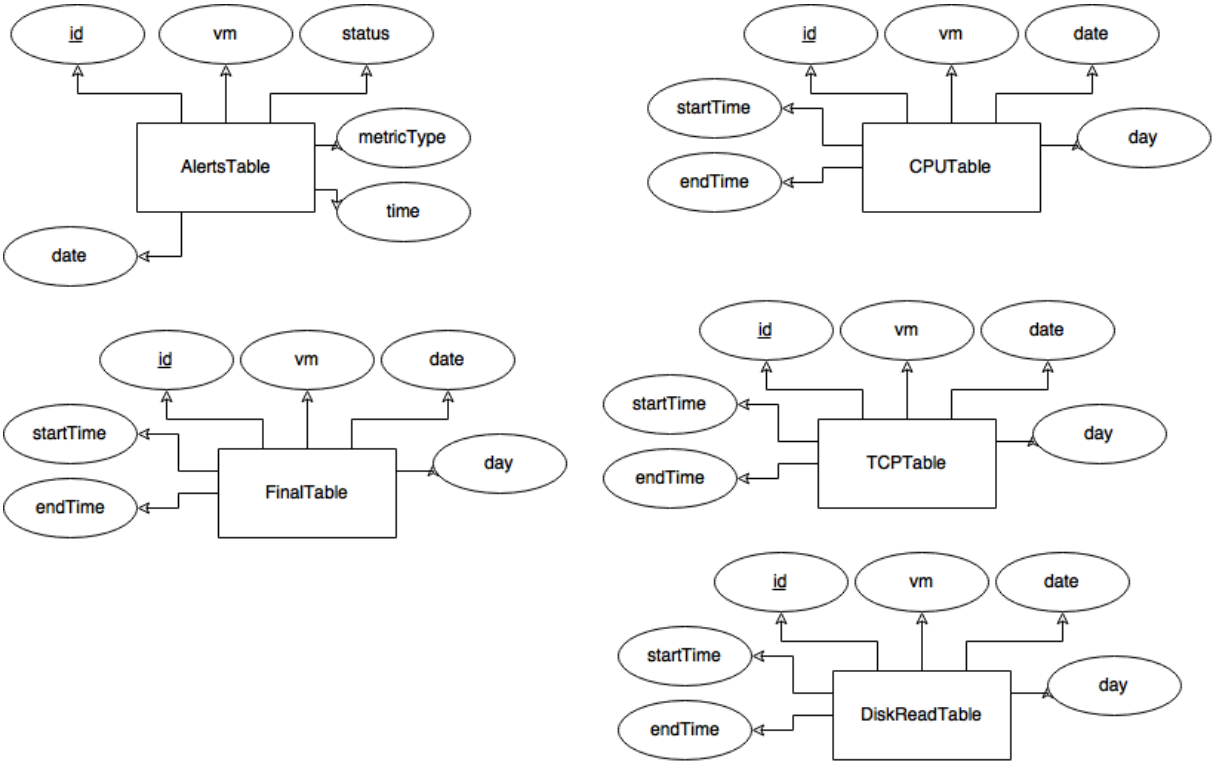


Bild 4.5. E-R diagram över databasen.

4.3 Modell i ML Studio

I ML Studio har mycket tid och fokus lagts på att träna algoritmer med data från databasen för att ta fram en prediktiv modell. Möjligheterna till att förbättra en modell finns men kräver mer tid. Två prediktiva modeller som förutser tider för påslagning respektive avstängning av virtuella maskiner har tagits fram. Modellerna är identiska med skillnaden att de tränas på antingen starttid eller sluttid.

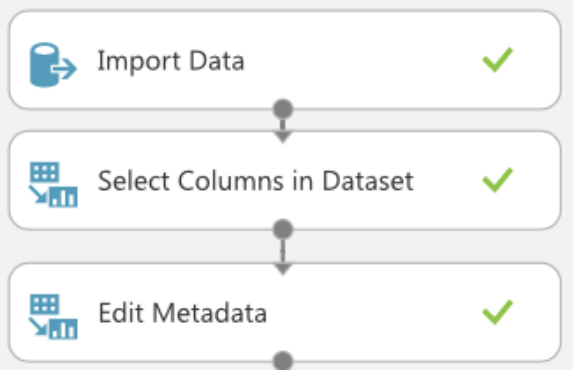
4.3.1 Manipulering av dataset

Med hjälp av det grafiska gränssnittet samt drag-och-släpp metoden i ML Studio är det enkelt att hämta, använda, ändra och manipulera dataset. Manipulering av dataset behöver göras för att skapa ett användbart dataset som skall träna en algoritm.

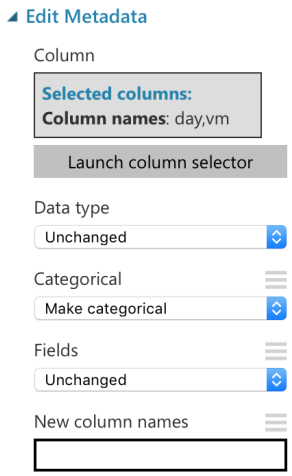
FinalTable i databasen skapades i åtanke om att ta med rätt data för att skapa en så bra prediktiv modell som möjligt. Ju fler kolumner (funktioner) som används för att träna maskininlärningsalgoritmer desto bättre prediktiv modell kan tas fram och därför används så många kolumner som möjligt. De kolumner som används är vm, date, day, startTime och endTime.

Kolumner som kan kategoriseras görs med Edit Metadata modulen. Kolumner som är icke-kontinuerliga måste kategoriseras då det är stor skillnad på matematiska metoder för kontinuerliga och icke-kontinuerliga värden. Kolumner som kategoriserats ses som val eller alternativ snarare än som ett numeriskt intervall [13]. Kolumner som kategoriseras är vm och day. Figur 4.6 visar de första stegen :

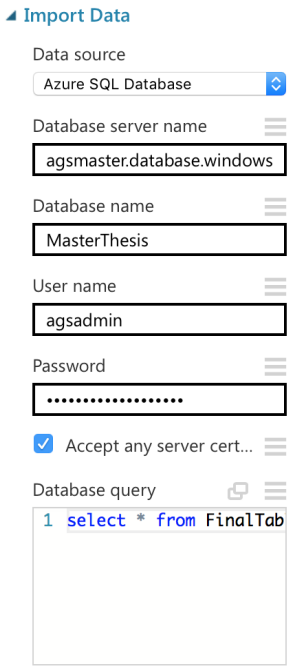
1. Dataset hämtas från databasen (bild 4.8)
2. Ej användbara kolumner tas bort (id och starttid/sluttid beroende på vilken som inte skall förutses)
3. Redigering av kolumner (bild 4.7), vm och day kategoriseras



Figur 4.6. De första stegen i ML Studio för experimenten som tar fram en modell vardera för start- och sluttid.



Figur 4.7. Kolumnerna day och vm kategoriseras.



Figur 4.8. hämtning av dataset från databasen.

4.3.2 Testning av algoritmer

När manipulering av dataset har gjorts skall den mest lämpade maskininlärningsalgoritmen väljas. Undersökningar har gjorts för att hitta en algoritm som tillsammans med datasetet ger bäst resultat. ML Studio tillhandahåller ett flertal algoritmer som är anpassade efter vad som skall förutses. Algoritmerna är uppdelade i fyra kategorier varav Classification kategorin är uppdelad i två underkategorier. Dessa är Two-Class-Classification och Multi-Class-Classification som förutser en kategori (kolumn) mellan 2 kategorier respektive mellan fler än 2 kategorier. Med hjälp av bild 4.9 och Microsofts webbsida för hur man väljer rätt algoritm [14] har undersökningar gjorts för att ta reda på vilken algoritm som är bäst lämpad för att förutse tider för påslagning och avstängning av en virtuell maskin.

Då en modell där en specifik kolumn skall förutses samt fler än två kolumner skall används för träning är Multi-Class den mest lämpliga kategorin att använda. Träningen av en Multi-Class-algoritm på datasetet fungerar på sådant sätt att det skapas ett givet antal beslutsfattande träd som tränas på data från datasetet för att sedan rösta på en kategori [15]. Dessa kategorier är olika tider.

Bild 4.10 visar ett test på hur Multi-Class Logistic Regression, Multi-Class Desicion Jungle och Multi-Class Desicion Forest (Multi-Class Neural Network-algoritmen samt One-V-All Multi-Class-algoritmen gav alldeles för dåliga värden och är därför inte med i testet) presterade jämfört med det verkliga värdet (Actual data) från datasetet. Testet utfördes på sådant sätt att starttiderna respektive sluttiderna som fanns i datasetet (Actual data) summerades, sedan tränades de tre algoritmerna vardera med samma dataset, resultatet summerades och jämfördes med det verkliga värdet. Algoritmen med minst skillnaden till det verkliga värdet är den som förutser bäst tider.

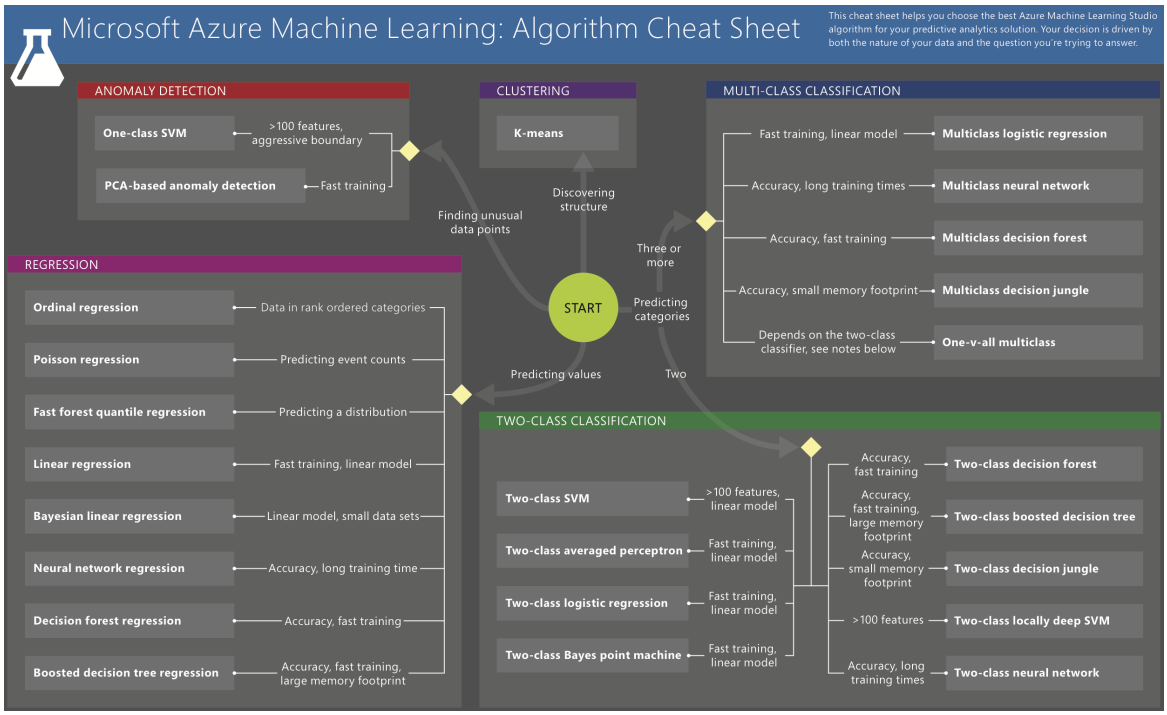


Bild 4.9. Fusklapp som hjälper dig välja rätt maskininlärnings algoritm.

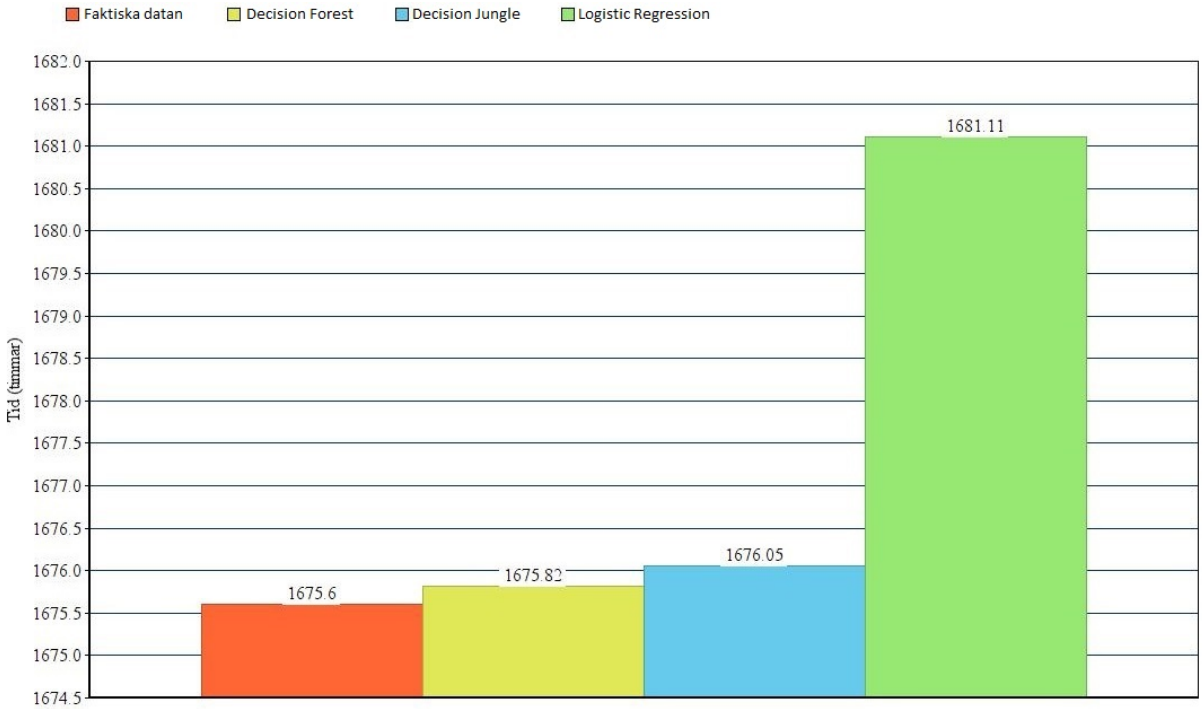


Bild 4.10. Graf över test av multi-klass algoritmer i timmar.

4.3.3 ML Studio modell som webb-service

När en prediktiv modell skapats är den fortfarande i experiment-stadiet och kan inte användas eller testas i applikationer.

Vid användning av webb-service kräver modellerna för starttid respektive sluttid parametrarna vm, day och date då det är dessa kolumner som har använts för att träna algoritmen. Svar skickas tillbaka med en starttid respektive sluttid. Bild 4.11 visar ett kodexempel på hur en JSON-begäran ser ut med avseende på värdena i Values som är de manuella inmatningar som används av modellen för att förutse en tid. Modellen i ML Studio tar emot JSON-objektet och svarar med ett JSON-objekt (se bild 4.12).

Uppdatering av systemarkitekturen visas i bild 4.13. Varningsregler har satts upp, en runbook har skapats, databas har formats och modeller har skapats i ML Studio som tar fram förutsägelser för start- och stopptider. All kommunikation fungerar mellan delarna förutom kommunikation som involverar applikationen då den återstår att skapas.

```
{
 "Inputs": {
  "input1": {
   "ColumnNames": [
    "vm",
    "day",
    "date"
   ],
   "Values": [
    [
     "agsVM",
     "6",
     "2016-05-27"
    ]
   ]
  }
 },
 "GlobalParameters": {}
}
```


Bild 4.11. JSON-begäran som skickas från applikation till webb-service.


```
{
 "Results": {
  "output1": {
   "type": "DataTable",
   "value": {
    "ColumnNames": [
     "Scored Labels"
    ],
    "ColumnTypes": [
     "Categorical"
    ],
    "Values": [
     [
      "17:04:00"
     ]
    ]
   }
  }
 }
}
```


Bild 4.12. JSON-objekt skickas som svar från modell till applikation.

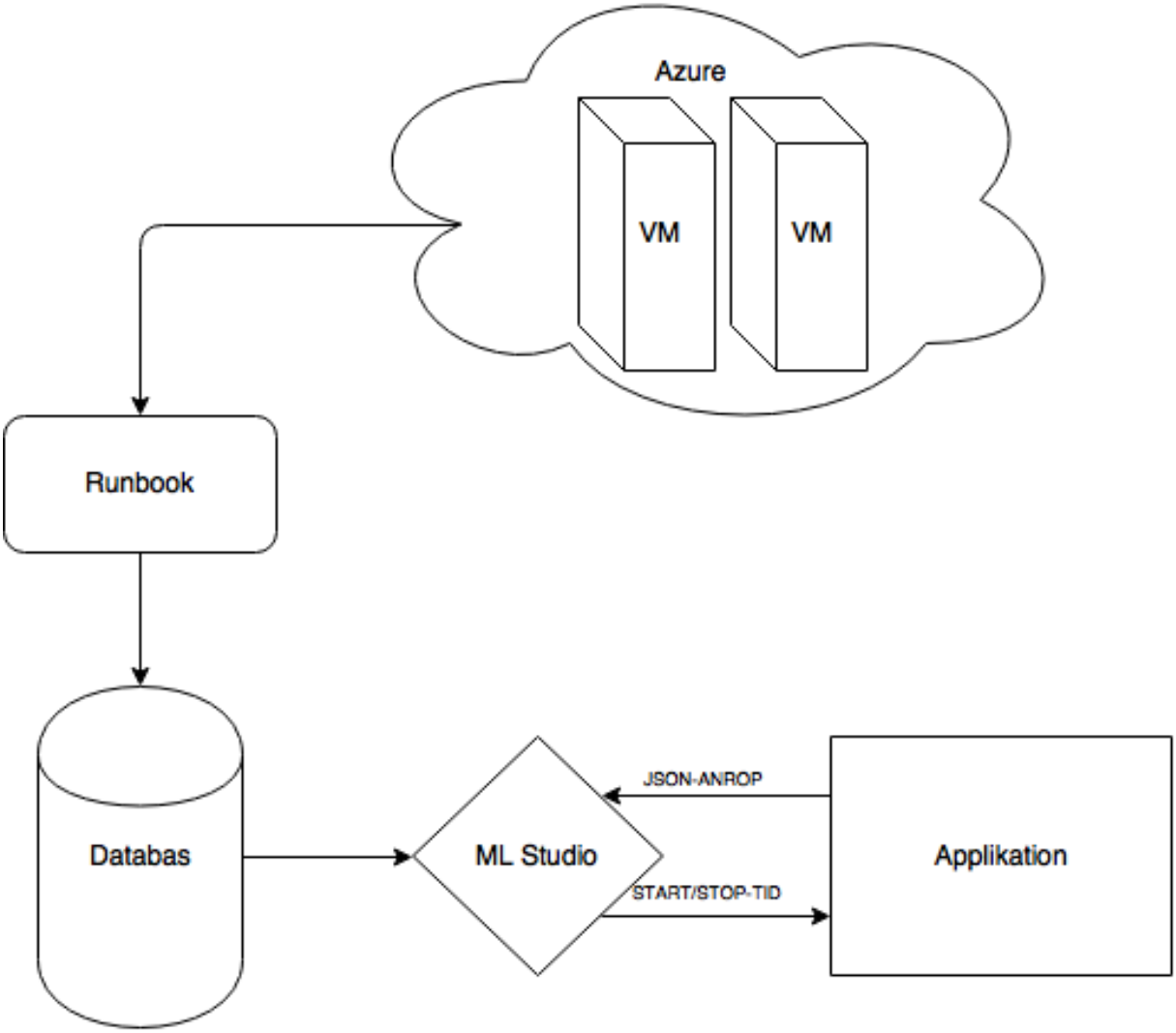


Bild 4.13. Uppdaterad Näringskedjan där applikationen återstår att skapas. Övriga delar är skapade och kommunikation sinsemellan fungerar.

4.4 Applikation

Applikationen agerar som "spindeln i nätet" då den hanterar alla virtuella maskiner som skall hanteras. Processen från att göra en JSON-begäran till ML Studio till att starta eller stoppa en virtuell maskin fungerar på följande sätt:

1. Varje dag görs ett anrop till ML Studio med förfrågan om startid och sluttid.
2. ML Studio returnerar en start- och sluttid och applikationen schemalägger en startoperation och en stoppoperation med dessa tider.
3. När starttid respektive sluttid inträffar skickas ett anrop till en runbook som antingen stänger av eller slår på en virtuell maskin.

Bild 4.14 visar ett UML-diagram för applikationen. VMmanager utför det mesta arbetet och Main skapar ett VMmanager-objekt för varje virtuell maskin som skall hanteras. StringTable används för metoden som kommunicerar med ML Studio. StringTable har också som uppgift att spara parametrarna som skall skickas till ML Studio.

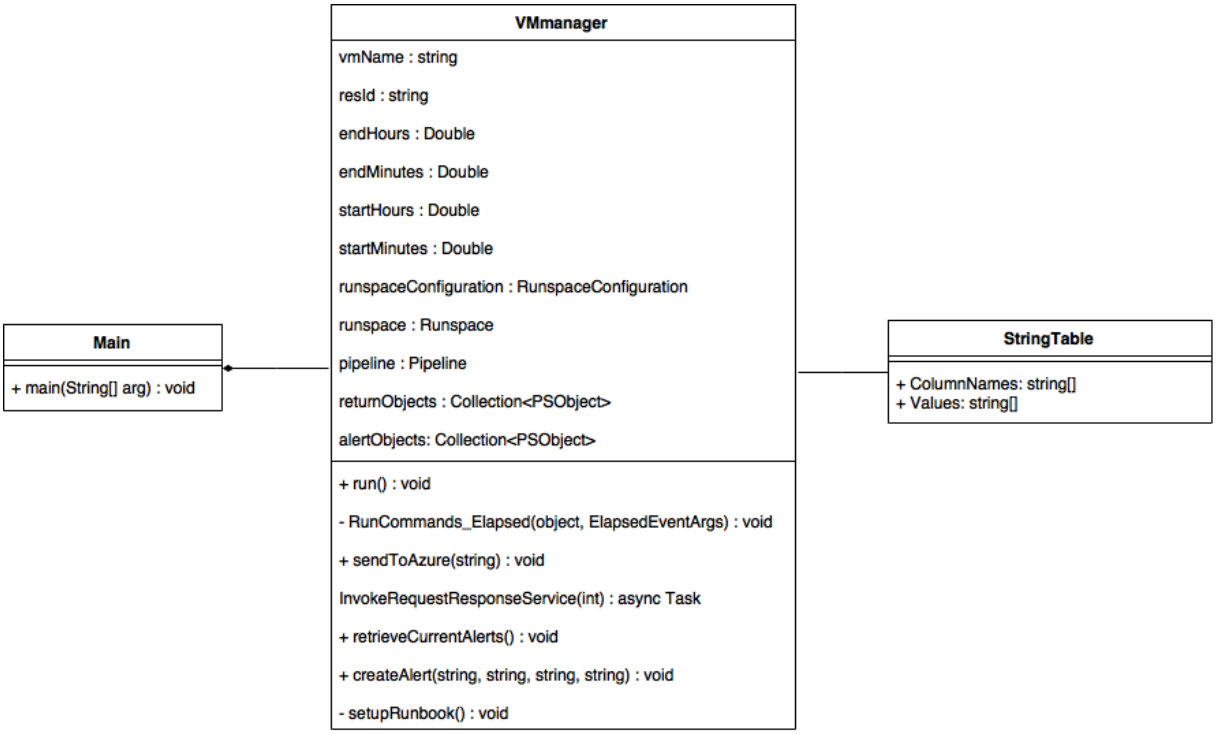


Bild 4.14. UML-diagram för applikation.

4.4.1 Operation med PowerShell

För att hantera de virtuella maskinerna används runbooks då PowerShell-skript är det primära tillvägagångssättet för att manövrera tjänster i Azure. REST-anrop[2] är ett annat smidigt tillvägagångssätt dock saknades stöd som behövdes för att kunna hantera en virtuell maskin.

Manövrering av tjänster i Azure kan utföras med hjälp av runbooks, dock inte hantering av varningsregler vilket är ett problem. Att ta emot data från varningsregler går, men att utföra funktioner som t.ex. att undersöka antal tillgängliga varningsregler eller skapa varningsregler saknas och dessa funktioner är nödvändiga i arbetet för att förenkla processen för utökning av virtuella maskiner. Utan dessa funktioner måste varningsregler läggas till manuellt för varje ny virtuell maskin som skapas vilket är tidskrävande. Problemet är att de runbooks som körs i Azure saknar ett PowerShell-tillägg och tillägget finns endast tillgängligt via nedladdning på en Windows-dator. Detta löses genom att PowerShell-skript körs via applikationen och datorn som kör applikationen måste installera detta PowerShell-tillägg.

[2] REST-anrop är ett sätt för två maskiner att kommunicera [16]

4.4.2 Anrop till runbook för att starta/stoppa en virtuell maskin

Ytterligare en runbook har skapats samt hanterar de virtuella maskinerna. Denna runbook är kopplad till en webhook och fångar JSON-objekt från applikationen med information om vilken virtuell maskin som skall hanteras och om denna skall startas eller stoppas.

Bild 4.15 visar koden för den runbook som tar emot information från applikationen och hanterar de virtuella maskinerna. Raderna 23 och 27 utför start- respektive stopkommandon på en given virtuell maskin.

```
1  param (
2      [object]$WebhookData
3  )
4  # If runbook was called from Webhook, WebhookData will not be null.
5  if ($WebhookData -ne $null) {
6      #Login to Azure Account
7      $PSCred = Get-AutomationPSCredential -Name 'Login'
8      Login-AzureRmAccount -Credential $PSCred
9      Select-AzureRmSubscription -SubscriptionName 'Visual Studio Premium med MSDN'
10
11     #Retrieve Resource Group Name and Automation Account Name from Assets
12     $ResName = Get-AutomationVariable -Name 'ResName'
13
14     # Collect properties of WebhookData
15     $Data = ConvertFrom-Json -InputObject $WebhookData.RequestBody
16     $VMName = $Data.vmName
17     $Operation = $Data.operation
18
19     Write-Output "You are operating on the following virtual machine: $VMName"
20
21     if($Operation -eq "start") {
22         Write-Output "Following virtual machine is being turned ON: $VMName ..."
23         Start-AzureRmVM -ResourceGroupName $ResName -Name $VMName
24     }
25     elseif($Operation -eq "stop") {
26         Write-Output "Following virtual machine is being truned OFF:  $VMName ..."
27         Stop-AzureRmVM -ResourceGroupName $ResName -Name $VMName -Force
28     }
29 }
30 else {
31     Write-Error "Runbook mean to be started only from webhook."
32 }
```

Bild 4.15. Runbook som hanterar anrop från applikationen och stänger av eller slår på en virtuell maskin.

5 RESULTAT

En applikation som kommunicerar med ML Studio har skapats. Med hjälp av MainRunbook hanterar applikationen också de virtuella maskinerna. Vi har lyckats hämta och spara användardata från de virtuella maskinerna för att sedan träna en algoritm i ML Studio. Modeller har skapats i ML Studio som förutser tider för påslagning och.

Bild 5.1 visar den slutgiltiga versionen av Näringskedjan som representerar de fem byggstenarna som arbetet är uppbyggt på. Samtliga byggstenar har implementeras och kommunikation sinsemellan dem är upprättad.

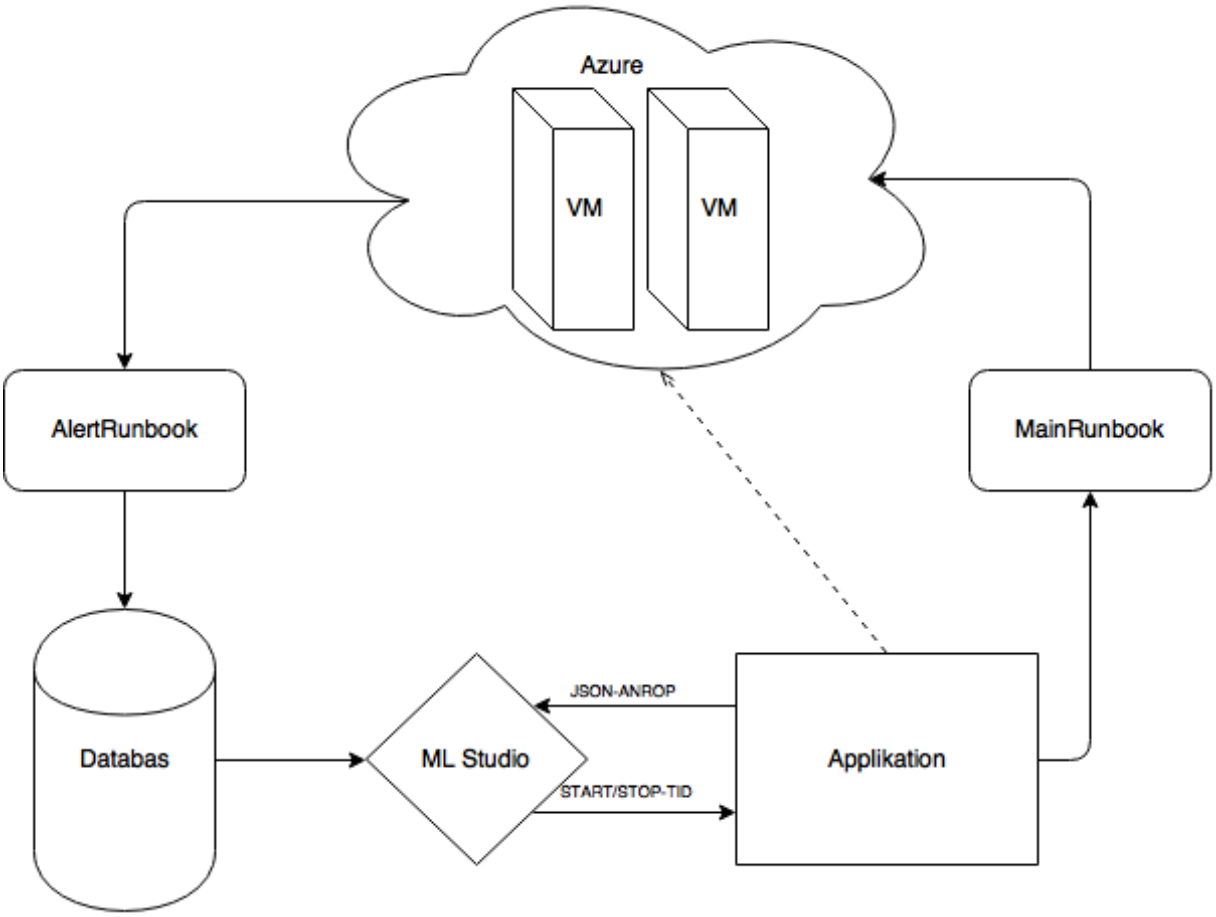


Bild 5.1. Slutgiltig version av Näringskedjan.

Arbetet kan delas upp i två delar:

Datasamlingsdel - Byggstenarna här är virtuella maskiner, AlertRunbook, Databas och ML Studio. Denna del av Näringskedjan används för att spara information om hur de virtuella maskinerna används i ett dataset. Detta måste göras för att kunna träna en algoritm i ML Studio och ta fram en förutsägels emodell som används av applikationen.

Applikationsdel - Byggstenarna här är Applikation, MainRunbook och ML Studio. Applikationen är den sammankopplade anslutningen mellan övriga byggstenar i applikationsdelen. Den upprättar förfrågning- och svarskommunikation med ML Studio för att få tider för påslagning och avstängning, skickar begäranden om att antingen starta eller stoppa en virtuell maskin till MainRunbook och vid behov skapar varningsregler för virtuella maskiner. Applikationen startas en gång och skall köras på en dator eller virtuell maskin utan att stängas av. Processen i applikationen sker i följande steg:

1. Om inga varningsregler är skapade för en virtuell maskin skapas först dessa.
2. Förfrågan om morgondagens starttid och sluttid skickas till ML Studio via webb-service.
3. En timer sätts igång som kontinuerligt var tionde minut kontrollerar om klockan är +/- 10 minuter från starttiden eller sluttiden.
4. Om så är fallet skickas en begäran om avstängning eller påslagning till MainRunbook.
5. Morgondagens starttid eller sluttid hämtas beroende på om en avstängning eller påslagning utförts.

De frågeställningar som ställts upp har blivit besvarade:

Vilka lämpliga metoder finns det för att undersöka användarmönstret för en virtuell maskin?
Svaret på frågan är med hjälp av varningsregler. Varningsregler fungerar som övervakning av ett mätvärde på virtuella maskiner och varnar när given tröskel för mätvärdet passerats.

Vad för användardata skall övervakas för att avgöra om en virtuell maskin används?
Mätdatan som varningsreglerna övervakar är CPU-, TCP- och disk-användningen. En kombination av dessa värden är tillräcklig för att ge bra svar på om en virtuell maskin används.

På vilket sätt skall användardatan hanteras för projektets ändamål?
Användardatan från varningsreglerna sparas i en databas. Databasen manipulerar denna data samt utför ytterliga handlingar för att förbereda datan för ML Studio.

Hur skall ML Studio användas för att effektivisera användningen av en virtuell maskin?
Med hjälp av ML Studio har två prediktiva modeller tagits fram. Dessa modeller förutser tider för påslagning och avstängning av virtuella maskiner och unika användarmönster för virtuella maskiner kan tas fram. Modellerna har gjorts tillgängliga för användning i applikationer där de utnyttjats genom svara med förutsedda tider för antingen start- eller stopptid baserat på virtuell maskin, dagnummer och datum.

Hur sker avstängning och påslagning av virtuella maskiner efter att ha fått resultat från ML Studio?
Uppsättning av webb-service görs på enstaka steg. Efter tider för påslagning och avstängning hämtats i applikationen sätts en timer igång som kontinuerligt var tionde minut kontrollerar om klockan är +/- 10 minuter från start- eller sluttiden. Om så är fallet skickas en begäran om avstängning eller påslagning till en runbook.

6. SLUTSATS

6.1 Resumé

Maskininlärning har blivit alltmer populär. Att kunna utnyttja maskininlärning för att spåra bankbedrägeri, förutse ett företags framtida intäkter, ta beslut om när en jetmotor behöver underhållas istället för att behöva göra det manuellt.

Målet har varit att undersöka möjligheten att optimera användningen av virtuella maskiner i Azure med hjälp av ML Studio. Detta har uppnåtts genom framtagning av användarmönster för virtuella maskiner och utifrån det stänga av och sätta på dem med hjälp av en applikation. Utöver skapandet av en applikation och en modell i ML Studio har även en databas implementeras.

6.2 Kritisk diskussion

Det kan uppstå tillfällen då en användare bryter sitt användarmönster vilket vi inte tagit hänsyn till som kan leda till oönskad avstängning av en virtuell maskin. Följande är ett exempel på vad som inträffa:
En virtuell maskin är förutsagd att stängas av kl.16.00. När klockan slår 16.00 sitter en användare på den virtuella maskinen och arbetar med ett dokument offline. Utan någon kontroll stängs den virtuella maskinen ner och all data som inte sparats går förlorat.

En bättre lösning för undersökning av tillgängliga varningsregler samt skapandet av varningsreglerna finns som möjlighet för vidareutveckling. Att vara tvungen att installera ett PowerShell-tillägg för att kunna köra applikationen är inte den optimala lösningen.

Ett önskemål är att skapa en webbapplikation istället för en applikation som körs via konsolen på datorn. Detta diskuterades med AGS men då vi saknade kunskaper i detta område samt att det inte var något som prioriterades högt av AGS blev valet en konsoll-applikation. Om webbapplikation gjorts hade Azure utnyttjas då det finns tjänster till det. Applikationen hade istället kunnat köras på webben vilket hade medfört att den varit smidigare att använda. En webbapplikation hade löst problemet med att behöva installera ett PowerShell-tillägg.

6.3 Reflektion

Mycket tid behövde spenderas på att få grundläggande kunskaper inom maskininlärning, ML Studio samt PowerShell-skript och programmering i C# med .NET-ramverket då kunskaper om detta saknades. Det har varit lärorikt, intressant och utmanande men också tidskrävande då mycket tid lagts ned på moment som inte behövt ta den tid den tagit. Att det skulle vara tidskrävande att skaffa kunskap inom samtliga delar var något vi visste från början och därför hade extra tid för det planerats in.

AGS hade inga specifika riktlinjer eller faktiska krav på vad som ansågs som ett färdigt arbete. På grund av detta samt att kunskap saknades i större delar var det svårt att uppskatta vilka moment som skulle kräva mer tid än andra vilket ledde till en tidsplanering som var svår att följa. AGS såg arbetet mer som en chans att ta reda på om det ens gick att optimera användningen av virtuella maskiner i Azure med ML Studio. De ville att vi skulle undersöka området och potentialen till att omvandla deras idé till verklighet. Trots detta lyckades en övergripande planering göras som mestadels också hölls hela vägen.

Flera diskussioner har ägt rum där fokus har legat på att ta reda på vad både vi och AGS ansåg vara en färdigt arbete. Det inträffade ett antal situationer där vi trott att vi hade kommit till ett färdigt stadie men efter möten själva eller tillsammans med handledarna på AGS kommit på bättre lösningar. Detta tillvägagångssätt försökte repeteras varje vecka för att på ett smidigt sätt kunna stämma av med företaget om de lösningar man gjort var bra eller behövde förbättras.

6.4 Vidareutveckling

Att kunna redovisa något som kan dra ner på kostnaderna för ett företag är något som drar åt sig uppmärksamhet. En översiktlig idé är att förbättra applikationen på sådant sätt att den blir smidig att implementera oberoende av vilket företag det handlar om samt att lösa problemet i avsnitt 6.2.

LITTERATURREFERENSER

[1] Maskininlärning, Wikipedia. Wikipedia, 18 Maj 2016, https://sv.wikipedia.org/wiki/Maskininlärning (Acc 2016-06-17)

[2] David Chappell: Introducing Azure Machine Learning - A guide for technical professionals, San Fransisco, California 2015

[3] Varun Chandola, Arindam Banerjee, Vipin Kumar: Anomaly Detection: A Survey, Minneapolis-Saint Puals, Minnesota 2009

[4] JSON, Wikipedia. Wikipedia, 30 Nov 2015 https://sv.wikipedia.org/wiki/JSON (Acc 2016-06-17)

[5] Getting Started With Azure Automation - Runbook Management, Beth Cooper. Microsoft, 19 Jun 2014, https://azure.microsoft.com/en-us/blog/azure-automation-runbook-management/ (Acc 2016-06-7)

[6] POST (HTTP), Wikipedia. Wikipedia, 20 Maj 2016, https://en.wikipedia.org/wiki/POST_(HTTP) (Acc 2016-06-17)

[7] What's a Webhook, Nick Quinlan. SendGrid Inc, 24 Jun 2014, https://sendgrid.com/blog/whats-webhook/ (Acc 2016-06-06)

[8] About Webhooks, Github Inc. Github Inc, 2016, https://help.github.com/articles/about-webhooks/ (Acc 2016-06-06)

[9] Microsoft Visual Studio, Wikipedia. Wikipedia, 24 Feb 2015 https://sv.wikipedia.org/wiki/Microsoft_Visual_Studio (Acc 2016-06-06)

[10] Welcome to Visual Studio 2015, Microsoft. Microsoft, https://msdn.microsoft.com/en-us/library/dd831853.aspx (Acc 2016-06-06)

[11] How to monitor Microsoft Azure virtuell maskins, John Matson. Datadog, 2016 https://www.datadoghq.com/blog/how-to-monitor-microsoft-azure-v,s/

[12] Zapier, Zapier Inc. Zapier Inc, 2016, https://zapier.com (Acc 2016-06-07)

[13] Building a predicitve model in Azure Machine Learning Studio, Data Science Dojo. Data Science Dojo, 2014 http://datasciencedojo.com/building-classification-model-azure-machine-learning-studio/

[14] "How to choose algoritmhs for Microsoft Azure Machine Learning", Brandon Rohrer, Microsoft, 28 Apr 2016, https://azure.microsoft.com/sv-se/documentation/articles/machine-learning-algorithm-choice/ (Acc 2016-06-7)

[15] Multiclass Decision Forest, Microsoft. Microsoft, 12 Mar 2016 https://msdn.microsoft.com/sv-se/library/azure/dn906015.aspx (Acc 2016-06-07)

[16] Representational State Transfer, Wikipedia. Wikipedia, 7 Maj 2016 https://sv.wikipedia.org/wiki/Representational_State_Transfer (acc 2016-06-07)

APPENDIX

Tidsplan

En uppskattad tid för hur lång tid olika delmoment kommer att vara svårt att förutspå redan nu men det kommer att finnas ungefärliga riktlinjer att följa i tio veckor för examensarbetet.

De första två veckorna kommer läggas på att lära sig och förstå Azure Machine Learning Studio eftersom det är det centrala biten är det viktigt att veta vad man kan åtstadkomma med det och hur man åstadkommer det. En del fokus kommer även att läggas på C# då det är det preliminära beslutet på språk som valts för att bygga detta program för datorn.

Under vecka 3 och 4 skall en klar bild finnas på hur problemet skall lösas samt med vilka metoder. Under vecka 4 skall programmet för datorn börja formas.

Under vecka 5 och 6 har vi kommit mer än halvvägs in i arbetet och då skall algoritmen bakas in i programmet. Vid slutet av vecka 6 skall den första prototypen vara klar och då bör programmet kunna hämta information om användarens data-aktivitetsmönster och mata in det i algoritmen som i sin tur svarar med när datorn bör stängas av.

Vecka 7 och 8 kommer fokus läggas på förbättring av algoritmen samt skall programmet finslipas och diverse funktioner och grafer skall läggas in. Vid slutet på vecka 8 skall programmet fungera bättre än första prototypen från vecka 6.

De två sista veckorna lämnas som andrum för mer finslipning samt att fixa diverse problem som man stött på i slutet av projektet.