Machine Learning för smartare pendling

JOHANNES SUNNANVÄDER
BACHELOR´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY |
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Machine Learning för smartare pendling

LUNDS UNIVERSITET
Campus Helsingborg

LTH Ingenjörshögskolan vid Campus Helsingborg
Institutionen för datavetenskap

Examensarbete:
Johannes Sunnanväder

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Lunds universitet
Lund 2018

Sammanfattning

Vid resor med kollektivtrafik finns det applikationer där en resenär kan söka efter resor. Resorna presenteras i en lista som oftast är sorterad efter avgångstid eller ankomsttid, vilket inte alltid stämmer överens med hur en resenär faktiskt väljer sina resor. Examensarbetet syftar till att utvärdera om det är möjligt att utveckla en maskininlärningsmodell som istället sorterar resorna utifrån resenärens användarhistorik.

För att undersöka om det är möjligt att utveckla en sådan maskininlärningsmodell har examensarbetaren genom användning av Amazons maskininlärningstjänst Amazon Machine Learning utvecklat en maskininlärningsmodell som förutspår vilka resor en resenär mest sannolikt kommer att välja utifrån en lista med resor. Maskininlärningsmodellen använder historisk data över en resenärs valda resor för att förutspå vilka av resorna i den presenterade listan resenären mest sannolikt kommer att välja.

I examensarbetet har de parametrar som maskininlärningsmodellen använder tagits fram, till exempel antal minuter till avgång och antal byten, och ett datafilsformat för att presentera dessa parametrar för maskininlärningsmodellen har skapats utifrån de framtagna parametrarna. Inom Amazon Machine Learning har även ett transformationsrecept tagits fram som ger maskininlärningsmodellen en hög precision (i genomsnitt 91 till 98 procent enligt testerna).

Som ett exempel på hur maskininlärningsmodellen fungerar i praktiken har en prototyp på en Android-applikation utvecklats. I prototypen kan en resenär söka efter och välja resor. Resorna sorteras med hjälp av maskininlärningsmodellen och presenteras sorterade utifrån sannolikheten att de kommer att väljas av resenären.

Resultatet av examensarbetet är en fungerande maskininlärningsmodell med en tillhörande prototyp på en Android-applikation där maskininlärningsmodellen används samt tester som visar att maskininlärningsmodellen fungerar.

Nyckelord: maskininlärning, Amazon Machine Learning, reseapplikation, pendlare, kollektivtrafik

Abstract

When traveling with public transport there is nowadays applications where a traveler can search for trips. The trips is presented as a list and is most often sorted by departure time or arrival time, which not always matches how a traveler actually chooses his trips. This thesis aims to test if it is possible to develop a machine learning model that instead sorts the trips by how a traveler has chosen trips in the past.

To try if it is possible to create such a machine learning model, the thesis worker has through the use of Amazon's machine learning service Amazon Machine Learning developed a machine learning model that predicts which trips a traveler most likely will choose from a list of trips. The machine learning model uses historic data over a travelers chosen trips to predict which of the trips in the presented list the traveler most likely will choose.

Through the thesis the parameters that the machine learning model uses has been extracted, for example number of minuter to departure and number of exchanges, and the format of the datafiles to present these parameters to the machine learning model have been created. In Amazon Machine Learning a transformation recipe has been created that gives the machine learning model as high precision as possible (on average 91 to 98 percent according to the tests).

As an example of how the machine learning model works in practice a prototype of an Android application has been developed. In the prototype it is possible to search for and choose trips. The trips is sorted with the help of the machine learning model and is presented sorted by the possibility that they are chosen by the traveler.

The result of the thesis is a working machine learning model with an associated prototype for an Android application where the machine learning model is used and also tests to prove that the machine learning model works.

Keywords: machine learning, Amazon Machine Learning, travel application, commuter, public transport

Förord

Det här examensarbetet gjordes möjligt av Martin Rosenqvist på tretton37. Jag vill rikta ett stort tack till honom och hans medarbetare och även min handledare på LTH, Christin Lindholm.

Jag har genom examensarbetet fått mycket nya kunskaper och utvecklats som både utvecklare och människa. Då i princip alla moment och tekniker i examensarbetet var nya för mig fick jag lära mig en massa nytt och utveckla min problemlösningsförmåga. Det har varit en rolig och lärorik upplevelse från början till slut, så återigen, tack så mycket!

Innehållsförteckning

1 Inledning

I kapitlet beskrivs bakomliggande information om examensarbetet såsom syftet och målformuleringen för examensarbetet.

1.1 Bakgrund

Examensarbetet har utförts i samarbete med konsultbolaget tretton37. Företaget grundades år 2010 i Lund och har idag cirka 200 anställda med kontor i Lund, Helsingborg, Stockholm, Borlänge, Östersund och Ljubljana i Slovenien.

tretton37 är ett konsultbolag som hjälper sina klienter att nå sina mål genom att leverera digitala produkter och tjänster. För att möjliggöra detta har företaget konsulter med olika specialistkompetenser såsom projektledare, utvecklare, designers och testare. tretton37 arbetar med olika typer av teknologier till exempel maskininlärning, mobil utveckling och webbutveckling. Exempel på klienter är Alfa Laval, Deutsche Bahn, Tetra Pak och Verisure.

Examensarbetet utförs mot en av tretton37s kunder inom kollektivtrafikbranschen. Då resenärer ofta har preferenser för hur de vill att deras resa ska vara, till exempel så få byten som möjligt, kortast möjliga färdtid eller önskemål på färdväg vill tretton37 utvärdera hur sökresultaten kan förbättras med hjälp av maskininlärning.

För att undersöka hur maskininlärning kan förbättra sökresultaten har en maskininlärningsmodell med tillhörande prototyp av en Androidapplikation utvecklats. Prototypen ger resenären relevanta resvägsförslag baserat på parametrar som tagits fram under examensarbetet. Maskininlärningsmodellen har integrerats på prototypens sökfunktion och använder de framtagna parametrarna för att sortera resultaten efter relevans för resenären. Efter examensarbetet kommer tretton37 vidareutveckla maskininlärningsmodellen som tagits fram för att sedan använda resultatet i deras reseapplikation som de utvecklar till en kund.

1.2 Syfte

Examensarbetet syftar till att testa om maskininlärning kan användas för att förbättra resupplevelsen för resenärer.

1.3 Målformulering

Målet för examensarbetet är en fungerande maskininlärningsmodell med tillhörande

prototyp av en Android-applikation. Maskininlärningsmodellen ska kunna förutse vilka resor som är mest relevanta för resenärerna och sortera resultaten efter detta.

1.4 Problemformulering

I detta examensarbete besvaras följande frågor:

1. Vilka parametrar finns det som kan påverka en resenärs val av resväg?
2. Vilken eller vilka transformationsrecept ska användas i modellen för att önskat resultat ska uppnås?
3. Hur ska man gå tillväga för att, utifrån data från maskininlärningsmodellen, sortera resultaten efter relevans?

1.5 Motivering av examensarbetet

Genom att förbättra sökresultaten för en reseapplikation ska pendlande bli enklare. Förhoppningen är då att man kan bidra till att fler personer väljer att pendla kollektivt istället för att ta bilen och i förlängningen förbättras miljön.

Examensarbetet valdes då förbättring i hur sökresultaten presenteras i en reseapplikation är en intressant lösning för resenärer. Examensarbetet är även intressant då det handlar om maskininlärning som har blivit en del av mjukvaruutveckling.

Företagets motivering är att de ser maskininlärning som ett viktigt område för branschen då det är ett kraftfullt verktyg som har många användningsområden. De ser en framtid för maskininlärning och AI inom mjukvarubranschen och deras företag och väljer därför att satsa på detta område.

Målgruppen för maskininlärningsmodellen som utvecklas i examensarbetet är resenärer som ofta använder sig av kollektivtrafiken, såsom resenärer som pendlar till och från jobbet eller skolan varje dag.

1.6 Avgränsningar

Prototypen kommer endast utvecklas för Android, delvis på grund av att examensarbetaren själv använder Android vilket gör testningen enklare, delvis på grund av att Android-applikationer kan skrivas i Java-kod vilket examensarbetaren har bra förkunskaper i.

Maskininlärningsmodellen kommer använda ett begränsat antal parametrar. Den kommer inte baseras på tidpunkt på dygnet, plats eller årstid. Avgränsningen beror på att tiden är begränsad och att examensarbetet endast utförs av en person.

2 Teknisk bakgrund

Det här kapitlet beskriver tekniker och teorier som har använts i examensarbetet.

2.1 Android Studio

Prototypen är skriven i Android Studio. Här utvecklas gränssnittet och logiken för prototypen. Android Studio är en integrerad utvecklingsmiljö som är utvecklad av Google för att skriva Androidapplikationer. Den baseras på Java eller Kotlin för logiken och XML för gränssnitten. I det här examensarbetet används Java.

Androidutveckling baseras på aktiviteter. En aktivitet representerar en enskild skärm för applikationen och består av en XML-fil för gränssnittet och en Java-fil för logiken och funktionaliteten. Aktiviteter hanterar användarevent och visar gränssnittet. Aktiviteter kan starta andra aktiviteter genom “intents”, vilket gör att den nya aktiviteten hamnar ovanför den föregående på skärmen. Det innebär att när en ny aktivitet startas försvinner inte den gamla utan läggs i bakgrunden.

När en aktivitet skapas anropas dess onCreate metod. I metoden måste man specificera vilken gränssnittsfil som ska användas för aktiviteten med metoden setContentView, samt binda gränssnittsobjekt, till exempel ett textfält, som applikationen kan behöva använda med metoden findViewById. I Figur 2.1 visas ett exempel på hur koden för att skapa en aktivitet med ett textfält och en knapp ser ut.

```
/* in activity_example.xml */
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ExampleActivity"
    android:orientation="vertical">

    <TextView
        android:id="@+id/example_text_view"
        android:layout_width="match_parent"
        android:layout_height="100px" />

    <Button
```

```
        android:id="@+id/example_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Click Here"/>
</LinearLayout>


/* in ExampleActivity.java */
public class ExampleActivity extends AppCompatActivity {
    TextView exampleTextView;
    Button exampleButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_example);

        exampleTextView = findViewById(R.id.example_text_view);
        exampleButton = findViewById(R.id.example_button);
    }
}
```

Fig. 2.1 - Exempel på hur en aktivitet utvecklas i Android Studio.

Android Studio erbjuder funktioner för att installera en applikation på en extern enhet. Det går även att skapa en virtuell enhet som körs på datorn och installera applikationen på den. Man kan sedan använda en debugger för testning av applikationen när den används, vilket underlättar felsökning och testning.

2.2 Maskininlärning

Maskininlärning innebär att man genom algoritmer tränar en modell som, givet en viss data, kan förutsäga en eller flera parametrar i ett dataset. Förutsägelsen baserar sig på historisk data som maskininlärningsmodellen har tränat på. Genom algoritmer hittar maskininlärningen mönster, matematiska funktioner och relationer mellan indata och utdata. Baserat på detta kan en förutsägelse göras för att givet indatan X bör utdatan vara Y.

Inom maskininlärning finns det flera olika kategorier av inlärningstekniker. De tre mest använda är övervakad inlärning, oövervakad inlärning och återkopplingsinlärning. (E. Alpaydin, 2010)

- **Övervakad inlärning:** Övervakad inlärning innebär att man redan från början har ett träningsset innehållande indata och tillhörande utdata. Maskininlärningsalgoritmerna kan då hitta samband mellan indatan och utdatan.

- **Oövervakad inlärning:** Oövervakad inlärning innebär till skillnad från övervakad inlärning att man har ett träningsset som endast består av indatan. Maskininlärningen grupperar sedan dessa baserat på vilka attribut de består av. Maskininlärningsalgoritmerna hittar då samband och likheter mellan indatan och placerar dessa i kategorier.
- **Återkopplingsinlärning:** Återkopplingsinlärning har, likt oövervakad inlärning, ingen fördefinierad utdata. I återkopplingsinlärning försöker maskininlärningen nå ett visst mål och vet att den är på rätt eller fel spår genom belöningar respektive bestraffningar. Den vanligaste användningen av detta är maskininlärning som ska lära sig att spela ett spel. Målet är att vinna spelet, så det är den största belöningen. Här kan det finnas delmål som ger lite mindre belöningar, såsom poäng eller hur långt i spelet man kommer. Den största bestraffningen är att förlora och mindre bestraffningar kan vara att man tar skada. Utifrån belöningar och bestraffningar lär sig maskininlärningsmodellen de bästa tillvägagångssättet för att vinna.

Det här examensarbetet använder övervakad inlärning då datan som samlas genom applikationen alltid innehåller svaret, det vill säga vilken resa resenären valde.

Processen för att skapa och använda en maskininlärningsmodell går till enligt Figur 2.2. Figuren är baserad på en artikel skriven av Yufeng Guo (Yufeng Guo, 2017).

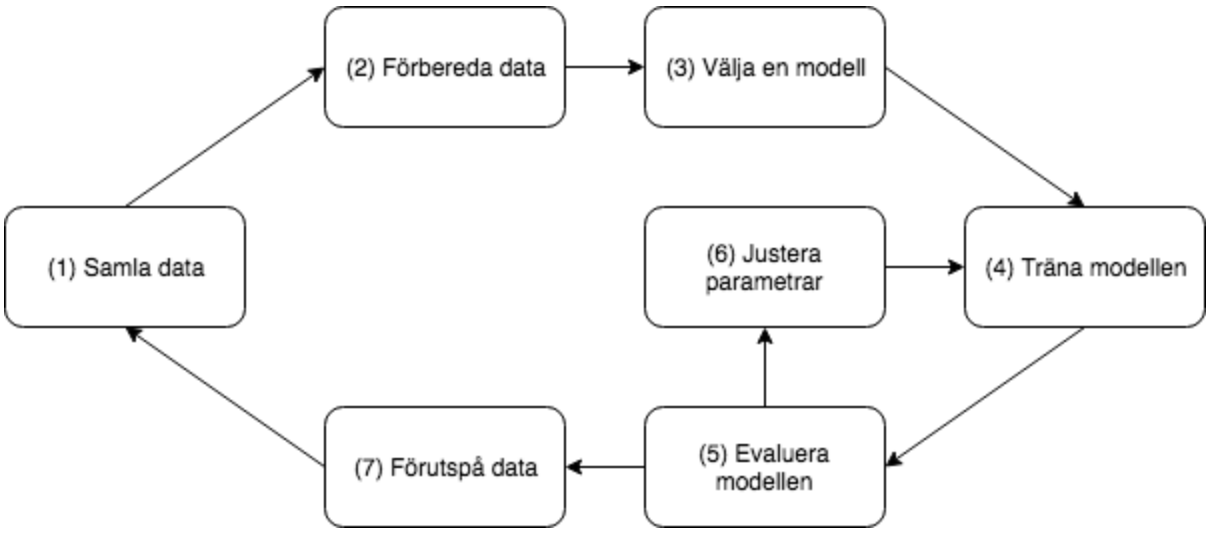


Fig. 2.2 - Processen för att skapa och använda en maskininlärningsmodell.

Figur 2.2 visar att maskininlärning är en iterativ process. Som steg 1 för att bygga en maskininlärningsmodell samlas data in för att träna maskininlärningsmodellen på. När datan är insamlad förbereds datan i steg 2 genom att de parametrar som ska användas formateras och presenteras på ett sätt som maskininlärningsmodellen kan dra slutsatser ifrån (Miroslav Kubat,

2017). Här delas även datan in i två delar, en del som blir träningsdata och en del som blir evalueringsdata.

När steg 1 och 2 är utförda måste typen av modell som ska användas väljas i steg 3. Vilken modell som ska användas baseras på vilken typ av data man har och vilken typ av problem som ska lösas. Därefter tränas maskininlärningsmodellen med hjälp av träningsdatan i steg 4. Under träningsprocessen hittar maskininlärningsmodellen de matematiska funktioner och mönster som beskriver relationen mellan indata och utdata.

När maskininlärningsmodellen är tränad evalueras modellen i steg 5 med hjälp av evalueringsdatan. I en evaluering undersöks maskininlärningsmodellens prestanda som ofta uttrycks i felfrekvens, det vill säga hur många förutsägelser av det totala antalet som gav fel resultat. I vissa fall kan det istället vara fördelaktigt att undersöka prestandan som maskininlärningsmodellens precision (antalet korrekta förutsägelser dividerat med totala antalet förutsägelser) eller "recall" (antalet sanna positiva förutsägelser för alla positiva värden i evalueringsdatan) (Miroslav Kubat, 2017). Baserat på resultatet från evalueringen kan det vara önskvärt att öka prestandan på maskininlärningsmodellen. Detta görs genom att i steg 6 justera maskininlärningsparametrarna för att bättre anpassa dessa till maskininlärningsmodellen. Ett exempel på en parameter som kan justeras är antalet iterationer som maskininlärningsmodellen ska göra över träningsdatan vid träning. Efter justering av maskininlärningsparametrarna går man igenom steg 4 och 5 igen och upprepar detta tills ett tillfredsställande resultat uppnåtts.

När ett tillfredsställande resultat har uppnåtts i en evaluering kan maskininlärningsmodellen användas för att förutspå utdata baserat på ny indata i steg 7. När en förutsägelse utförts sparas datan undan för att sedan användas för att träna om maskininlärningsmodellen baserat på den nya datan som tillkommit.

2.3 AWS

Amazon Web Services (AWS) är en plattform som innehåller en samling molntjänster utvecklade av Amazon såsom lagring, maskininlärning, beräkning, dataanalys och webhosting. AWS erbjuder servrar i 19 regioner utspridda över världen. Det finns fyra i Europa, sju i Asien, en i Oceanien, sex i Nordamerika och en i Sydamerika (Amazon, u.å). Att servrarna finns i olika regioner bidrar till lägre svarstid men högre tillgänglighet för användarna.

2.3.1 S3

AWS Simple Storage Service (S3) är en av de molnlagringstjänster som AWS erbjuder. I S3 lagras objekt såsom textfiler, bilder eller videos. Objekten lagras i så kallade "buckets".

En “bucket” är den behållare där objekt lagras. För en bucket kan det specificeras vilka användare som ska kunna skapa, ta bort och lista objekten i en “bucket”. Regionen för var en “bucket” ska skapas och objekten ska lagras kan även specificeras. I en “bucket” finns det ingen gräns för hur många objekt som kan lagras.

2.3.2 Amazon Machine Learning

Amazon Machine Learning är en molnbaserad maskininlärningstjänst där maskininlärningsmodeller på ett enkelt sätt kan utvecklas. Genom Amazon Machine Learning kan tre typer av maskininlärningsmodeller utvecklas: binär klassificeringsmodell, multinomial klassificeringsmodell och regressionsmodell.

En binär klassificeringsmodell förutser ett binärt utfall, det vill säga ett utfall som kan ha två värden, med hjälp av en algoritm som heter “logistic regression”. En multinomial klassificeringsmodell förutser ett utfall som kan anta flera värden inom ett begränsat område med hjälp av algoritmen “multinomial logistic regression”. En regressionsmodell förutser ett numeriskt värde med hjälp av algoritmen “linear regression”. (Amazon Web Services, u.åa)

Regression innebär ett försök att hitta en relation mellan ett antal input-variabler och en output-variabel som beror på input-variablerna genom att att hitta en matematisk funktion för input-variablerna (E. Alpaydin, 2010). Den matematiska funktionen tas fram baserat på hur mycket påverkan input-variablerna har på värdet för output-variabeln. “Logistic regression” använder regression för att tilldela en poäng mellan 0 och 1 till två output-variabler. Poängen representerar sannolikheten att variabeln kommer vara den som väljs. “Multinomial logistic regression” fungerar på liknande sätt som “logistic regression” med skillnaden att sannolikheten fördelas på ett flertal output-variabler. “Linear regression” använder regression för att hitta en matematisk funktion som beräknar ett konkret värde baserat på input-variablerna.

För att träna en maskininlärningsmodell används en datakälla. Datakällan skapas utifrån en csv-fil som ligger i Amazon S3. Under utvecklingen specificeras vilken typ av data som varje kolumn representerar, såsom “numeric”, “categorical” eller “binary”. Även kolumnen som maskininlärningen ska förutse specificeras.

När en maskininlärningsmodell utvecklas läses datakällan som ska användas som träningsdata in. Ett recept över hur datan ska transformeras inför inlärningsprocessen ska sedan utvecklas.

Recept skrivs i JSON-format och består av tre sektioner: “Groups”, “Assignments” och “Outputs”. I “Groups” kan variabler grupperas med varandra för att förenkla transformering på dessa. I “Assignments” kan variabler skapas för den transformerade datan om detta behövs. I

“Outputs” definieras vilka variabler inlärningsprocessen ska använda och vilka transformeringar som ska ske på dessa (Amazon Web Services, u.åa). Två viktiga transformationer som används i examensarbetets maskininlärningsmodell är “Quantile Binning” och “Cartesian Product”. “Quantile Binning” innebär att numeriska variabler delas in i kategorier baserat på variabelns värde. Antal kategorier specificeras i receptet, från 5 till 1000 kategorier. Kategorierna kommer sedan bestå av variabler inom ett visst tröskelvärde. “Cartesian Product” binder ihop två variabler till en eller flera strängar baserat på hur många ord (delsträngar separerade med whitespace) variablerna innehåller. “Cartesian Product” skapar delmängder av storleken två mellan två strängar. Delmängderna består av ett ord från varje sträng sammanbundna med understreck. En delmängd skapas för varje kombination av ord mellan strängarna (Amazon Web Services, u.åa). Om “Cartesian Product” appliceras på strängarna “cartesian product” och “quantile binning” ges strängarna {“cartesian_quantile”, “cartesian_binning”, “product_quantile”, “product_binning”}.

I Figur 2.3 visas ett exempel på ett recept taget från AWS egna dokumentation (Amazon Web Services, u.åa).

```
{
  "groups": {
    "LONGTEXT": "group_remove(ALL_TEXT, title, subject)",
    "SPECIALTEXT": "group(title, subject)",
    "BINCAT": "group(ALL_CATEGORICAL, ALL_BINARY)"
  },
  "assignments": {
    "binned_age" : "quantile_bin(age,30)",
    "country_gender_interaction" : "cartesian(country, gender)"
  },
  "outputs": [
    "lowercase(no_punct(LONGTEXT))",
    "ngram(lowercase(no_punct(SPECIALTEXT)),3)",
    "quantile_bin(hours-per-week, 10)",
    "hours-per-week,
    "cartesian(binned_age, quantile_bin(hours-per-week,10)",
    "country_gender_interaction",
    "BINCAT"
  ]
}
```

Fig. 2.3 - Ett exempel på ett transformationsrecept.

När man har skapat ett recept kan ett antal parametrar justeras. Följande parametrar kan justeras (Amazon Web Services, u.åa):

- **Maximal storlek på maskininlärningsmodellen:** Definierar hur stor maskininlärningsmodellen maximalt får vara. Om ett högt värde sätts och maskininlärningen inte kan hitta tillräckligt med mönster för att fylla storleken skapas en maskininlärningsmodell med mindre storlek.
- **Maximalt antal iterationer över träningsdatan:** Definierar hur många iterationer över träningsdatan maskininlärningen maximalt får göra för att hitta mönster. Om inga fler mönster kan hittas används inte alla iterationer.
- **Regulariseringstyp:** För att undvika att maskininlärningsmodellen anpassar sig för mycket för träningsdatan, så den endast presterar bra på exakt den data som finns med i träningsdatan och därmed presterar dåligt på övrig data kan regularisering behövas. Regularisering straffar de funktioner som hittas som antingen har låg eller hög vikt. Tre typer av regularisering kan väljas: ingen regularisering, L1 eller L2. L1 minskar antalet funktioner i modellen genom att funktioner med låg vikt får vikten noll. L2 resulterar i mindre vikter, vilket ger stabilare vikter.
- **Regulariseringsmängd:** Här justerar man mängden av regularisering som ska användas. Man kan välja tre olika nivåer: “mild”, “medium” eller “aggressive”. Om “mild” används kommer regulariseringen endast ha en liten påverkan på slutresultatet och endast påverka funktioner med extrema vikter, både höga och låga. “Medium” ger en lite större påverkan på slutresultatet och “aggressive” har en stor påverkan på slutresultatet genom att påverka alla funktioner, störst påverkan på funktionerna med extrema vikter och lite mindre påverkan på funktioner med normala vikter.

I nästa steg kan en evaluering av maskininlärningsmodellen skapas. Man kan välja om datasetet man ska använda för evaluering ska vara en del av träningsdatan eller om man vill använda ett annat dataset. Vid evaluering kan man se hur bra maskininlärningsmodellen presterar. Resultatet presenteras som ett diagram där antalet korrekta förutsägelser och felaktiga förutsägelser visas och ett tröskelvärde som visar precisionen för maskininlärningsmodellen.

2.3.3 Amazon API Gateway

Amazon API Gateway är en tjänst som möjliggör konnektivitet mellan tjänster i AWS och externa applikationer och system. Genom Amazon API Gateway skapas ett API som tilldelas en bas-URL. Utifrån bas-URL:et skapas API-metoder med med tillhörande väg-parameter som sedan binds till önskad funktion i AWS. För en API-metod kan man sedan lägga till HTTP-headers, auktorisationsfunktioner och statuskoder för HTTP-svaren.

I examensarbetet används Amazon API Gateway för att skicka och hämta data från S3 och för att skapa och uppdatera maskininlärningsmodellerna. Förutsägelser genom maskininlärningsmodellen sker med Amazon Machine Learnings Java-paket.

2.4 JSON

JSON står för JavaScript Object Notation och är en av standarderna för att lagra och skicka data mellan server och applikation. Ett JSON-objekt är textbaserat och består av nyckel-värde par. Giltiga typer på värdena är tal, sträng, boolean, array, JSON-objekt och null. Ett JSON-objekt som beskriver en användare kan se ut som i Figur 2.4.

```
{
   “userName” : “kalle”,
   “password” : “hemligt123”,
   “address” : {
      “street” : “Streetname”,
      “streetNumber” : “1a”,
      “zipcode” : 12345,
      “city” : “Helsingborg”,
      “country” : “Sweden”
   },
   “userId” : 1
}
```

Fig. 2.4 - JSON-Objekt för att beskriva en användare

2.5 API

API är en akronym som står för “Application Programming Interface”. Ett API används för att på ett enkelt sätt skicka data mellan två olika applikationer eller webbtjänster. Ett API består ofta av funktioner som nås genom att anropa en bas-URL följt av motsvarande väg-parameter. Man kan även lägga till data i anropet genom fråge-parametrar bestående av nyckel-värde par. Genom att anropa webbadressen för API:et med rätt väg-parameter och data får man ett svar. Svaret består av en statuskod som visar om anropet var lyckat eller misslyckat och i de fall data önskas finns även den här. Ett API-anrop till domänen “https://www.exempel.com/” med funktionen “login” och med användarnamnet kalle och lösenordet hemligt123 ser ut som i Figur 2.5:

```
https://www.exempel.com/login?username=kalle&password=hemligt123
```

Fig. 2.5 - Exempel på ett API-anrop

2.6 Gson

Gson är ett Java-bibliotek som används för att omvandla från JSON-objekt till Java-objekt eller från Java-objekt till JSON-objekt. För att omvandla ett objekt till eller från ett Java-objekt behövs en klass som innehåller attribut för den omvandlade datan. (Google, u.å)

Gson skapar ett JSON-objekt utifrån Java-objektet genom att omvandla variablernas namn till JSON-nycklar och variablernas värde till JSON-värden. Om variabeln ska omvandlas under ett annat namn annoteras den med @SerializedName(“namn”). Ett exempel på omvandling från ett Java-objekt till ett JSON-objekt visas i Figur 2.6.

```
public class User {
   public String userName;
   public String password;
   @SerializedName(“userId”)
   public int id;

   public User(String uName, String pass, int userId) {
      userName = uName;
      password = pass;
      id = userId;
   }
}

// main method
User user = new User(“kalle”, “hemligt123”, 1);
Gson gson = new Gson();
String json = gson.toJson(user);
/* json kommer nu vara
{“userName” : “kalle”, “password” : “hemligt123”, “userId” : 1} */
```

Fig. 2.6 - Exempel på omvandling från ett Java-objekt till ett JSON-objekt med Gson

Omvandling från JSON-objekt till Java-objekt visas i Figur 2.7.

```
String json = “{\”userName\” : \”kalle\”, \“password\” : \”hemligt123\”, \”userId\”
: 1}”;
Gson gson = new Gson();
User user = gson.fromJson(json, User.class);

/* user har nu värdena “kalle”, “hemligt123” och 1 som userName, password
respektive id */
```

Fig. 2.7 - Exempel på omvandling från ett JSON-objekt till ett Java-objekt med Gson

2.7 Retrofit 2

Retrofit2 är ett Java-bibliotek utvecklat av Square. Retrofit2 är en HTTP-klient som förenklar API-anrop inom Android genom att omvandla ett API till ett Java-interface. När data hämtas från ett API omvandlas detta till ett Java-objekt med hjälp av en JSON-omvandlare, exempelvis Gson (Square, u.å.).

För att använda Retrofit 2 behöver man huvudsakligen tre klasser:

- En Java-klass som ska instansieras av JSON-omvandlaren.
- Ett interface som representerar de möjliga API-anropen
- En Retrofit.Builder-klass. Den här klassen bygger en konkret klass utifrån interfacet.

Varje metod i interfacet måste Java-annoteras med motsvarande HTTP-metod, till exempel “GET”, “PUT” eller “POST”, baserat på vilken typ av HTTP-anrop som representeras samt vilken väg-parameter i API:t anropet ska använda. För att skicka med data i ett anrop används annoteringen @Body för parametern som ska skickas med och för att lägga till fråge-parametrar till anropet används annoteringen @Query(“key”) för parametern. I Figur 2.8 visas ett exempel på ett Retrofit 2 interface med metoderna login() och createUser().

```
public interface API {
   @POST(“login”)
   Call<LoginStatus> login(@Query(“userName”) String userName,
                           @Query(“password”) String password);
   @PUT(“createUser”)
   Call<Void> createUser(@Body User user);
}
public class LoginStatus {
    boolean accepted;
    int statusCode;
}
```

Fig. 2.8 - Exempel på ett Retrofit 2 interface med två metoder, login och createUser. Metoden login kommer returnera ett objekt av typen LoginStatus.

I Retrofit.Builder-klassen definieras en bas-URL, vilken JSON-omvandlare som ska användas, vilken typ av felhanterare som ska användas och övrig logit som retrofit 2 ska använda. Man använder sedan create()-metoden för att returnera en konkret klass av API-interfacet. Ett exempel på en statisk metod som skapar och returnerar en API-klass utifrån interfacet i Figur 2.8 visas i Figur 2.9.

```
public class APIBuilder {
    public static API getAPI() {
        Retrofit retrofit = new Retrofit.Builder()
                .baseUrl(“http://www.exampleapiurl.com/api/”)
                .addConverterFactory(GsonConverterFactory.create())
                .build();
        return retrofit.create(API.class);
    }
}
```

Fig. 2.9 - Exempel på en metod som skapar en konkret klass utifrån ett API-interface med Retrofit.Builder.

2.8 RxJava

RxJava är ett Javabibliotek skapat av ReactiveX. RxJava möjliggör asynkrona och event-baserade nätverksanrop med hjälp av "Observer"-mönstret. Genom att använda RxJava blir hantering av trådar abstraherat och koden blir enklare att läsa och förstå, till skillnad från om man använder Javas eller Androids inbyggda nätverkshantering.

Då RxJava bygger på "Observer"-mönstret (ReactiveX, u.å.) möjliggör detta en mer responsiv utformning av en applikation genom att applikationen kan reagera och presentera resultatet när datan har hämtats. Man slipper då avbrott i applikationens flöde under tiden data hämtas från exempelvis ett API-anrop.

RxJava består av fem olika klasser för hantering av nätverksanrop. Dessa klasser är:

- **Observable:** En klass som hanterar nätverksanrop som kan returnera en serie av värden. Klassen kan till exempel användas då applikationen ska visa status för en nedladdning genom en förloppsindikator. Observable kan då returnera hur många procent av nedladdningen som har blivit klar vid en viss tidpunkt.
- **Single:** Hanterar nätverksanrop som endast returnerar ett värde. Klassen kan användas när man hämtar data ifrån ett API-anrop.
- **Completable:** Hanterar nätverksanrop som inte returnerar något värde. Kan användas när data ska läggas in i en databas utan att något svar krävs.
- **Maybe:** Hanterar nätverksanrop som antingen returnerar ett eller inget svar. Kan användas för att se om något existerar i en databas. Om objektet existerar hämtas det annars hämtas inget.
- **Flowable:** Hanterar nätverksanrop som, precis som för Observable, kan returnera en serie av värden. Skillnaden från Observable är att man kan hantera större mängder data åt gången än vad Observable kan hantera genom att en strategi för hur överflöd av data ska hanteras. Överflöd sker när mer data tas emot än vad Observable kan göra av med.

Det är enkelt att kedja ihop eller parallellexekvera flera nätverksanrop med hjälp av RxJavas inbyggda funktioner. En sådan funktion är flatMap(), som anropas på en Observable. Metoden returnerar sedan en ny Observable när föregående Observable har hanterats klart. På det här viset kedjas nätverksanropen så att värdet från föregående anrop kan användas i nästkommande anrop.

Vid ett nätverksanrop prenumererar applikationen på resultatet med hjälp av metoden subscribe(). I subscribe() definieras en callback-metod där resultatet från nätverksanropet kan användas.

RxJava kan med fördel användas med Retrofit 2. Genom att i Retrofit.Builder klassen definiera RxJava som dess “Call Adapter” kan man ersätta returtypen för API-anropen från “Call<T> till någon av RxJavas klasser, exempelvis Observable<T>. I Figur 2.10 visas ett exempel på hur RxJava kan användas med Retrofit 2 samt hur kedjade nätverksanrop kan hanteras.

```
public interface API {
    @POST(“login”)
    Observable<Response<LoginStatus>> login(@Query(“userName”) String
userName,
                            @Query(“password”) String password);

    @PUT(“createUser”)
    Observable<Response<Void>> createUser(@Body User user);
}

// main method
API api = APIBuilder.getAPI();
User user = new User(“kalle”, “hemligt123”, 1);

api.createUser(user)
.flatMap(voidResponse -> {
    return api.login(user.userName, user.password);
})
.observeOn(AndroidSchedulers.io())
.subscribeOn(Schedulers.io())
.subscribe(loginStatusResponse -> {
    if(loginStatusResponse.body().accepted) {
        System.out.println(“User signed in”);
    } else {
        System.out.println(“User not signed in”);
    }
}
```

Fig. 2.10 - Exempel på hur RxJava kan användas tillsammans med retrofit 2 för att skapa och logga in en användare.

3 Metod

I det här kapitlet beskrivs arbetsprocessen som har använts i examensarbetet samt de olika faserna som examensarbetet har varit uppdelat i.

3.1 Arbetsprocess

Som arbetsprocess för examensarbetet användes en modifierad version av den agila utvecklingsmodellen Scrum (D. McKenna, 2016). Många av modifieringarna berodde på att examensarbetet utfördes av en person och Scrum är utvecklat för grupper. Anledningen till valet av agil arbetsprocess är att planeringen snabbt kan anpassas efter förändringar. Med regelbundna veckoplaneringar och retrospektiv har det varit lätt att följa hur examensarbetet har fortlöpt. Det underlättade för handledaren på företaget att följa upp arbetet och ge tips. Genom att för varje utfört delmoment skriva ner vilka lärdomar som kan dras och efter varje vecka skriva ett retrospektiv gavs även tillfälle att analysera arbetet och utveckla arbetsprocessen.

Examensarbetet har bestått av sprintar på en vecka. I början av varje vecka planerade examensarbetaren vad som skulle utföras under kommande sprint och skrev ner detta i en agenda tillsammans med en beskrivning över när det som skulle utföras kan anses vara klart. Under veckans gång följdes agendan och när en uppgift från agendan blev klar markerades detta som färdigt och en anteckning skrevs innehållande vilka lärdomar som kunde dras utifrån processen att lösa uppgiften, såsom om det tog längre eller kortare tid än planerat och varför det gjorde det eller om det fanns något som bör ha gjorts innan man började med detta. I slutet av varje vecka skrevs även lärdomar ner för det som inte blev färdigt. Det skrevs även en kort analys över vilka tekniker/arbetssätt som examensarbetaren skulle fortsätta använda, sluta använda och börja med att använda, en form av retrospektiv inför kommande vecka. Examensarbetaren fick då en överblick över vad som kunde användas för att förbättra arbetstakten i examensarbetet. I början var dessa dokument bristfälliga. Under examensarbetets gång utvecklades dokumenten genom feedback från handledaren på företaget och den personliga utvecklingen hos examensarbetaren i form av insikter och lärdomar och fick bättre struktur och mer givande innehåll.

En naturlig modifiering av Scrum när det endast var en person som utförde examensarbetet var att det inte fanns några roller eller någon Scrum-master. Dagliga “stand-ups” var även borttagna och ersattes av egna dagliga reflektioner över vad som utfördes föregående dag och vad som ska göras den aktuella dagen. Som hjälpmedel till detta användes en Scrum-board i Trello. Retrospektiven utfördes genom att efter varje sprint reflektera över hur väl målen för sprinten har uppnåtts och sedan skriva ner vilka lärdomar som kunde dras utifrån sprinten.

Kommunikation med företaget skedde genom möten med handledaren på företaget. Dessa skedde med en till två veckors mellanrum. På mötena gicks retrospektivet och agendan från föregående vecka igenom och progressionen på applikationen och maskininlärningsmodellen diskuterades. Även idéer om vad som kunde göras för att förbättra och utveckla prototypen diskuterades samt vad som skulle göras under nästkommande sprint.

Vid examensarbetets start fanns det inga specifika arbetstider att jobba utifrån utan arbete skedde när som helst under dygnet. Då detta ledde till att mindre arbete utfördes och att arbetet kunde ske sent på kvällarna infördes istället arbetstider som skulle följas. Arbetstiderna var mellan 08.00 till 16.00, med två raster på 30 minuter. Genom att göra detta undveks sena arbetsdagar, produktiviteten ökade och strukturen på examensarbetet förbättrades.

3.2 Faser

Det finns tre faser för examensarbetet. Dessa är informationsinsamling, utveckling och rapportskrivning. Varje fas har ett antal steg. I Figur 3.1 visas faserna med sina steg och hur dessa hänger ihop.

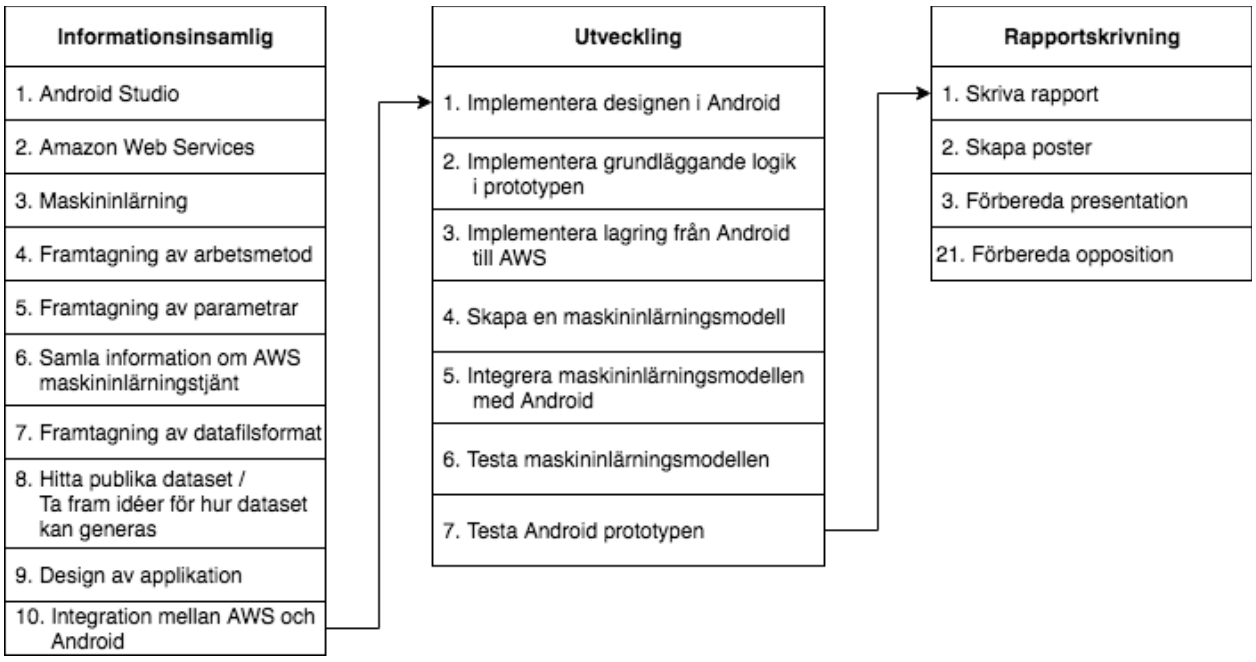


Fig. 3.1 - Visualisering av faserna samt deras steg

Examensarbetet var planerat att utföras under 16 veckor. Informationsinsamlingsfasen bestod av sex veckor, utvecklingsfasen bestod av sex veckor och rapportskrivningsfasen bestod av fyra veckor.

3.2.1 Informationsinsamligsfas

I informationssamlingsfasen samlades information om de olika teknikerna som kommer användas och utifrån det planerades applikationen.

Som steg 1 i informationssamlingsfasen samlades information om hur Android-utveckling sker i Android Studio. Genom att följa guider inhämtades kunskaper om hur detta fungerade och vilka nya programmeringstekniker Android-utveckling kräver utöver den Java-kunskap examensarbetaren redan besatt. En informativ guide var Udacitys "Android Developer Guide" (Udacity, u.å). Här fick examensarbetaren lära sig Android-utveckling från grunden med hjälp av videos och tester.

Steg 2 innebar att hitta information över vilka tjänster Amazon Web Services kunde erbjuda som skulle vara användbara i examensarbetet. När ett antal tjänster hade identifierats analyserades dessa genom att läsa igenom tjänsternas dokumentationer för att välja ut en tjänst för maskininlärning och en tjänst för lagring. En översiktlig uppfattning över hur dessa tjänster fungerar och hur de integreras i Android Studio skapades genom att läsa igenom deras dokumentationer.

I steg 3 skapades en inblick i hur maskininlärning fungerar genom att läsa artiklar, sammanfattningar och böcker. En bok som gav mycket information var Introduction to Machine Learning (E. Alpaydin, 2010). Genom att anteckna nyckelprinciperna för maskininlärning och skriva en sammanfattning för det som hade lästs sparades informationen undan för att användas under examensarbetets gång.

För att skapa en struktur i arbetsflödet togs det i steg 4 fram en arbetsmetod för examensarbetet. Arbetsmetoden togs fram med hjälp av handledaren på företaget och förbättrades sedan under examensarbetets gång med hjälp av feedback från handledaren och personliga lärdomar hos examensarbetaren.

När en arbetsmetod var framtagen och information över hur maskininlärning fungerar var insamlad skulle det i steg 5 tas fram parametrar som skulle sparas undan från de valda resorna och användas i maskininlärningsmodellen. Genom diskussion med handledaren på företaget togs så många parametrar som möjligt fram. Diskussionen följdes av en analys av examensarbetaren på parametrarna där orimliga eller irrelevanta parametrar togs bort. Efter analysen utfördes en djupare analys av examensarbetaren på kvarstående parametrar för att plocka ut några specifika parametrar som den version av maskininlärningsmodellen som skulle användas i examensarbetet kunde använda.

I steg 6 samlades kunskap in över hur Amazon Machine Learning fungerar. Genom att läsa dokumentationen för tjänsten uppnåddes en grundläggande förståelse. För att få en djupare förståelse skapades en testmodell med hjälp av ett publikt tillgängligt dataset enligt Amazons egen guide. Utifrån testmodellen skapades fler maskininlärningsmodeller med samma dataset men med olika parametrar och recept.

Genom att i steg 6 tagit reda på hur Amazon Machine Learning vill att datan ska presenteras skulle det i steg 7 tas fram ett datafilsformat för att presentera datan för Amazon Machine Learning. Först bestämdes det av examensarbetaren vilken typ av maskininlärningsmodell som verkade mest passande för ändamålet baserat på vilket resultat som behövdes från maskininlärningsmodellen samt vilka parametrar som skulle skickas in till maskininlärningsmodellen. Sedan skapades ett datafilsformat baserat på parametrarna som togs fram i steg 5. Datafilsformatet ändrades under examensarbetets gång, vilket går att läsa i kapitlet “3.2.2 Utvecklingsfas”.

För att kunna träna en maskininlärningsmodell behövdes ett dataset med data om olika resenärers resvanor. Därför utfördes det i steg 8 en sökning efter publika dataset över resenärers resvanor men inget hittades. Idéer om hur man istället kunde generera egna dataset som efterliknade verklig data så mycket som möjligt började tas fram. Genom att först planera vilken data som behövde samlas in utifrån resorna och sedan planera hur den skulle presenteras för att visa att en resenär valde eller inte valde en viss resa skissades en specifikation över ett program som genererar dataset med valda och inte valda resor upp.

I steg 9 skapades en pappersprototyp över hur gränssnitten i Android prototypen skulle se ut. Pappersprototypen presenteras i Appendix C.1.

Genom att i steg 10 läsa på om integration mellan AWS och Android bildade examensarbetaren en uppfattning över hur man kan spara undan och läsa in användarfiler mellan AWS och Android och även hur man skapar och använder en maskininlärningsmodell i AWS genom Android.

3.2.2 Utvecklingsfas

Som steg 1 av utvecklingsfasen skapades en grundläggande design av applikationen utan någon funktionalitet. Designen baserades på skissen som togs fram i steg 9 i informationsinsamlingsfasen och presenteras i Appendix C.2

I steg 2 implementerades den grundläggande logiken i applikationen. För att göra detta behövdes en klass som representerar resor. Därför skapades en Trip-klass. Klassen innehåller avgångs- och slutstation, avgångs- och ankomsttid, resvägen där alla byten och väntetider representerades samt variabler för de parametrar som skulle sparas undan till maskininlärningsmodellen. För att hämta in verkliga resor skapades en klass som anropar Google Directions API och skapar Trip-objekt utifrån resultatet som placeras i en lista. Klassen används sedan av applikationens sökfunktion.

För att spara undan data till maskininlärningsmodellen skapades det i steg 3 en klass som tar hand om API-anrop för att skicka filer till och från AWS S3. Klassen innehåller statiska metoder för att läsa in och skriva filer. Metoderna används på funktionen som sparar undan resor. En fil läses in, uppdateras med det nya resorna och sparas sedan igen.

I steg 4 skapades en maskininlärningsmodell. För att kunna skapa en maskininlärningsmodell behövdes ett dataset att testa på. Genom att följa specifikationen i steg 8 i informationsinsamlingsfasen skapades ett program som genererade sådana filer utifrån det framtagna datafilsformatet. Programmet genererar ett träningsset och ett evalueringsset baserat på förinställda preferenser. Genom programmet genererades fyra dataset med olika preferenser.

Ett transformationsrecept till maskininlärningsmodellen utvecklades. De olika transformationernas påverkan på precisionen testades först. Genom att testa med olika värden, parametrar och kombinationer av transformeringar byggdes kunskap upp om vad som gav bättre och sämre precision. Inga tester på det initialt framtagna datafilsformatet gav tillfredsställande precision.

På grund av att inget test gav tillfredsställande precision utfördes därför en analys för att hitta nya datafilsformat. Analysen baserades på hur man kunde presentera datan för maskininlärningsmodellen på ett så informativt och koncist sätt som möjligt. Det viktigaste var att maskininlärningsmodellen skulle kunna hitta tydliga mönster i datan. Genom analysen togs ett nytt datafilsformat fram. Vid tester på detta format med olika recept steg precisionen markant, men det blev fortfarande inte tillräckligt precist. Ännu ett nytt datafilsformat togs fram och efter justeringar i receptet uppnåddes en tillfredsställande precision.

Under en diskussion med handledaren på företaget föreslogs en idé om att kombinera maskininlärningsmodeller som behandlar dataset innehållandes olika data. Tanken var att detta skulle ge bättre resultat för resenären genom att kombinera olika typer av data. Diskussionens resultat blev att tre maskininlärningsmodeller skulle användas, en som hanterar ett litet dataset som innehåller en resenärs 100 senaste datarader (motsvarar cirka 15 resor) vars syfte är att snabbt reagera på förändringar i en resenärs preferenser, en som hanterar ett stort dataset som innehåller en resenärs alla föregående resor vars syfte är att ge en grund baserat på de preferenser

en resenär har haft under längst tid, och en som hanterar ett globalt dataset som innehåller alla resor för alla resenärer med syftet att ge en överblick över vilka preferenser de flesta resenärer har.

Då de tre maskininlärningsmodellerna ska påverka resultatet på olika sätt behövdes vikter baserat på hur mycket de ska påverka resultatet. För att ta fram vikterna bestämdes först vilken maskininlärningsmodell som skulle ha störst påverkan på resultatet. Detta bestämdes till den lilla maskininlärningsmodellen, då man vill att förändringar i en resenärs preferenser snabbt ska visas i resultatet. Då den lilla maskininlärningsmodellen använder ett dataset med cirka 15 resor kommer förändrade preferenser vara en majoritet i datasetet redan efter cirka 8 resor. Den globala maskininlärningsmodellen valdes till den maskininlärningsmodell som ska ha minst påverkan på resultatet. Detta då den ska användas för att skilja på resor som de övriga maskininlärningsmodellerna anser vara ungefär lika bra matchningar för en resenär med hjälp av hur preferenserna för en majoritet av alla resenärer ser ut. Vikterna sattes initialt enligt följande:

- Lilla maskininlärningsmodellen: 50%
- Stora maskininlärningsmodellen: 30%
- Globala maskininlärningsmodellen: 20%

Efter tester på maskininlärningsmodellerna med dessa vikter drogs slutsatsen att den lilla maskininlärningsmodellen behövde en högre vikt då resultatet inte visade förändringar i preferenser på ett korrekt sätt, samt att den globala maskininlärningsmodellen behövde en lägre vikt då den påverkade resultatet för mycket. Efter modifiering av vikterna sattes de till följande:

- Lilla maskininlärningsmodellen: 65%
- Stora maskininlärningsmodellen: 25%
- Globala maskininlärningsmodellen: 10%

Efter tester på maskininlärningsmodellerna med dessa vikter drogs slutsatsen att dessa vikter gav önskvärt resultat.

När maskininlärningsmodellerna utvecklats var det dags för att i steg 5 integrera Amazon Machine Learning med Android. En klass skrevs som möjliggör förutsägelser genom maskininlärningsmodellerna i Android. För detta användes Amazon Machine Learnings Android paket.

Utöver att anropa maskininlärningsmodellerna för förutsägelser behövdes metoder för att uppdatera maskininlärningsmodellerna vid förändring i en resenärs dataset. För att göra detta skapades en bakgrundstjänst med hjälp av Androids “JobService”. Bakgrundstjänsten körs i bakgrunden oavsett om applikationen är igång eller inte. Var åttonde timme kontrollerar den om resenären har valt en ny resa. Om så är fallet uppdateras resenärens maskininlärningsmodeller samt den globala maskininlärningsmodellen. För att uppdatera modellerna används Amazon API Gateway.

När maskininlärningsmodellerna och Android-prototypen utvecklats anordnades en kodgranskning på tretton37 av handledaren på företaget. Personen som granskade koden var en konsult som jobbade på företaget. Under kodgranskningen poängterades det att det fanns Android-paket som kunde hantera API-anropen på ett smidigare sätt, Retrofit2 och RxJava. Det fanns även ett paket som tolkar JSON-svaren från API-anropen och skapar klasser av detta, Gson.

De nya paketen som användes efter kodgranskningen tillämpades i applikationen. För att göra detta skapades först Java-klasser som ska ta emot API-svaren. Sedan skapades två "builder class" och två interfaces, en för Google Directions API och en för Amazon API Gateway. Interfacet för Amazon API Gateway innehåller arop för alla funktioner som behövs från AWS.

Det var nu dags för steg 6 och 7 i utvecklingsfasen, att testa maskininlärningsmodellerna och Android-prototypen. Två testdokument skapades, ett för applikationen och ett för maskininlärningsmodellerna. Testdokumentet för applikationen skrevs i "test case" format med förkrav, testinstruktioner, förväntat slutresultat och checkboxar för om testet lyckades eller inte. Det fanns även plats för en kommentar. Testdokumentet för maskininlärningsmodellerna skrevs i instruktionsformat, där varje del som skulle testas beskrevs steg för steg. Maskininlärningsmodellerna testades i två steg. De första testerna testade tröskelvärden för maskininlärningsmodellerna för olika dataset. De andra testerna testade hur resultaten presenteras i prototypen. Resultaten för de första testerna bestod av tröskelvärdena för evalueringen av maskininlärningsmodellerna presenterade i en tabell. Resultaten för de andra testerna bestod av en bild på hur resultaten presenteras i applikationen samt en observation av om resultaten matchar det förväntade.

För att utföra tester på maskininlärningsmodellerna genererades dataset som representerade valda resor för fyra resenärer med olika preferenser. Varje resenär hade tre dataset var, ett litet dataset med 100 rader (cirka 15 resor), ett mellanstort dataset med 400 rader (cirka 60 resor) och ett stort dataset med 1000 rader (cirka 150 resor). Det fanns även ett globalt dataset som innehöll en kombination av alla resenärers stora dataset.

Preferenserna för resenärerna var uppdelade enligt följande, där preferens 1 har hög vikt och preferens 2 låg vikt:

Resenär 1:

1. Distans
2. Minuter till avgång

Resenär 2:

1. Tid vid byten
2. Minuter till avgång

Resenär 3:

1. Antal byten
2. Minuter till avgång

Resenär 4:

1. Minuter till avgång
2. Gångavstånd

För varje resenär genererades även ett evalueringsset bestående av 2500 rader. Varje evalueringsset bestod av ny genererad data med samma preferenser som resenärerna.

Maskininlärningsmodellernas precision testades först genom evaluering i Amazon Machine Learning. Maskininlärningsmodeller skapades med hjälp av resenärernas dataset och evaluerades sedan mot resenärernas evalueringsset. Precisionen för de olika maskininlärningsmodellerna antecknades i en tabell. Tabellen presenteras i kapitel 5.4.

Nästa steg i testningen var att testa hur resultaten presenteras i prototypen och om det stämmer överens med examensarbetarens förväntningar baserat på resenärens preferenser. För att göra detta valdes först två hållplatser att söka resor mellan, från Pankow, Berlin till Messe Berlin. Den tidigaste avgångstiden för sökresultaten valdes till 12.00. En sökning utan maskininlärning gjordes och en skärmdump togs på sökresultaten. Sökresultaten numrerades sedan för att enklare kunna referera till dom i testerna.

Testning av maskininlärningsmodellerna bestod av tre typer av tester:

- Tester för resenärer med preferenser som inte ändras
- Tester för resenärer som har bytt preferenser
- Tester under tiden en resenär byter preferenser

Under testerna för resenärer med preferenser som inte ändras användes den globala maskininlärningsmodellen, resenärens personliga stora maskininlärningsmodell och resenärens personliga lilla maskininlärningsmodell. Det utfördes sedan tre tester för varje resenär, ett där samtliga maskininlärningsmodeller användes, ett där endast den stora och den lilla maskininlärningsmodellen användes och ett där endast den lilla maskininlärningsmodellen användes.

Under testerna för resenärer som har bytt preferenser användes den globala maskininlärningsmodellen, resenärens personliga stora maskininlärningsmodell och en annan resenärs lilla maskininlärningsmodell. Här utfördes två olika tester för varje resenär, ett där

samtliga maskininlärningsmodeller användes och ett där endast den stora och den lilla maskininlärningsmodellen användes.

För de sista testerna, där det testas hur resultaten presenteras under tiden en resenär byter preferenser, skapades dataset där det lilla datasetet för resenär 3 succesivt omvandlades till det lilla datasetet för resenär 4. Totalt skapades fyra nya dataset, där första datasetet bestod av 80 rader från det lilla datasetet för resenär 3 och 20 rader från det lilla datasetet för resenär 4. Det andra datasetet bestod av 60 rader från det lilla datasetet för resenär 3 och 40 rader från det lilla datasetet för resenär 4, och så vidare fram till det sista datasetet som bestod av 20 rader från det lilla datasetet för resenär 3 och 80 rader från det lilla datasetet för resenär 4. För varje nivå utfördes tre olika tester, ett där samtliga maskininlärningsmodeller användes, ett där endast den stora och den lilla maskininlärningsmodellen användes och ett där endast den lilla maskininlärningsmodellen användes.

Efter varje test togs en skärmdump på resultaten och en kommentar skrevs för hur bra resultaten stämde överens med vad examensarbetaren förväntade sig baserat på resenärens preferenser. Skärmdumparna och kommentarerna presenteras i kapitel 5.4 samt Appendix B.3.

3.2.3 Rapportskrivningsfas

I rapportskrivningsfasen skrevs rapporten, en poster skapades och en presentation förbereddes. Här förbereddes även en opposition.

3.3 Källkritik

I examensarbetet har information om maskininlärning främst hämtats från böcker funna med hjälp av LUBSearch. Följande böcker har använts:

- D. McKenna, 2016. The Art of Scrum
- E. Alpaydin, 2010. Introduction to Machine Learning. 2 uppl.
- Miroslav Kubat, 2017. An Introduction to Machine Learning. 2 uppl.

Böckerna är utgivna av stora och kända bokförlag (Apress, The MIT Press samt Springer) och författarna är sakkunniga professorer med akademisk bakgrund, vilket gör böckerna till säkra källor att hämta information från.

En artikel angående maskininlärning (Yufeng Guo, 2017) har även lästs. Artikeln är publicerad på en välkänd hemsida (Towards Data Science) som är en del av Medium.com. På hemsidan publiceras artiklar främst av personer med akademisk bakgrund inom respektive ämne med syfte att lära ut, vilket gör hemsidan trovärdig. Artikelns författare är “Developer Advocate” på

Google med bakgrund som utvecklare. Då artikeln är författad av en person med goda kunskaper inom ämnet på en trovärdig hemsida kan slutsatsen dras att detta är en trovärdig källa.

Från Amazon Web Services officiella hemsida har tre källor använts. Dessa är följande:

- Amazon, u.å, Global Cloud Infrastructure
- Amazon Web Services, u.åa, Amazon Machine Learning
- Amazon Web Services, u.åb, Amazon Sagemaker

Amazon Web Services används av ett stort antal utvecklare och tillhör ett av världens största företag. Utifrån detta kan en slutsats dras om att det som anges i källorna stämmer.

ReactiveX, u.å är den officiella hemsidan för RxJava. RxJava är en välkänd teknik som används av många utvecklare och genom sökning efter fristående guider för tekniken hittades flera hemsidor som bekräftar informationen som står på hemsidan. Detsamma gäller för källan Square, u.å som är den officiella hemsidan för Retrofit2.

Google, u.å är den officiella användarguiden för Gson. Den är publicerad på Googles officiella GitHub-sida, i deras “repository” for Gson. Det är en teknik som används av många utvecklare och informationen i användarguiden bekräftas av ett flertal fristående sidor. Då Gson är publicerat på Googles officiella GitHub-sida och att ett flertal fristående sidor bekräftar informationen kan källan anses vara trovärdig.

BVG-Motion, u.å, är en hemsida som beskriver ett projekt som utförs av BVG och MotionTag. BVG är Berlins kommunala lokaltrafikbolag och Tysklands största lokaltrafikbolag. På deras officiella hemsida skriver de om projektet och hänvisar till källan som används i det här projektet. Då projektet beskrivs på deras officiella hemsida kan man anse att källan är trovärdig.

4 Analys

I kapitlet analyseras viktiga frågeställningar inom examensarbetet såsom vilka parametrar som ska användas i maskininlärningsmodellen och vilket transformationsrecept som ger bäst precision på maskininlärningsmodellen.

4.1 Val av maskininlärningstjänst

Vid val av vilken maskininlärningstjänst från AWS som skulle användas i examensarbetet stod valet mellan Amazon Machine Learning och Amazon SageMaker. Båda tjänsterna erbjuder möjligheten att skapa maskininlärningsmodeller. Skillnaden mellan dessa är att Amazon SageMaker har fler funktioner och erbjuder fler möjligheter att själv styra över maskininlärningsprocessen än vad Amazon Machine Learning gör.

I Amazon SageMaker skrivs kod för hur maskininlärningsprocessen ska se ut. Här kan data filtreras ut, modifieras och transformeras innan den används för att träna en maskininlärningsmodell. Amazon SageMaker kommer med flera förskrivna maskininlärningsalgoritmer som kan användas under träningsprocessen. Användaren kan även använda självskrivna maskininlärningsalgoritmer. När en maskininlärningsmodell är tränad kan maskininlärningsmodellen evalueras och sedan distribueras.

Amazon Machine Learning ger inte användaren samma frihet som Amazon SageMaker. Man kan till exempel inte modifiera träningsdata på samma sätt som i SageMaker. Filtrering och modifiering av data ska ske innan den skickas in till tjänsten och transformering begränsas till de sju alternativ som tjänsten erbjuder. Maskininlärningsalgoritmen bestäms av tjänsten baserat på vilken typ av data som ska förutsägas, såsom binär data eller linjär data.

Amazon Machine Learning valdes för att tjänsten är enklare att lära sig och kändes tillräcklig för att lösa examensarbetet med goda resultat.

4.2 Identifiering av parameterar

Identifiering av parametrar började med en diskussion med handledaren på företaget. Vid diskussionen låg fokus på vilka parametrar som kan påverka en resenärs reseupplevelse. Diskussionen följdes av en analys av vilka parametrar som är rimliga att använda i en maskininlärningsmodell. I analysen studerades data som är möjlig att hämta in från Google Directions API och utomstående källor såsom SMHI och utifrån detta valdes de parametrar som går att spara undan utifrån den datan. Efter analysen kvarstod parametrarna som hittas i Appendix B.1.

Parametrarna som framkommit kan delas in i två grupper, direkta parametrar och indirekta parametrar. Direkta parametrar är de som beskriver en resa, såsom antal byten, restid och gångavstånd. Dessa parametrar hämtas ifrån informationen om en resa. Indirekta parametrar beskriver saker utanför resan såsom väder, sevärdheter och om det finns något event i staden. Dessa parametrar kan förändra vilken resväg en resenär väljer då det påverkar omständigheterna kring resan. Som exempel kan vädret ha en påverkan på preferenserna genom att resenären inte bryr sig om väntetider och gångavstånd en solig dag men de dagar det är oväder spelar dessa preferenser en större roll.

De kvarstående parametrarna behövdes nu begränsas ytterligare för att kunna hanteras inom ramen för examensarbetet. En djupare analys i två steg utfördes för att begränsa parametrarna. Som ett första steg utfördes en analys på JSON-svaret från Google Directions API. Under analysen studerades vilka parametrar som gick att hämta ifrån JSON-svaret och utifrån informationen togs de indirekta parametrarna som kräver data från källor utanför Google Directions API bort. Som nästa steg identifierade examensarbetaren vilka parametrar som ansågs ha störst påverkan på en resenärs val av resa samt vilka parametrar som maskininlärningsmodellen kunde hitta mönster för baserat på resenärens val av resa. Då det inte fanns någon insamlad resdata baserades identifikationen på examensarbetarens personliga resvanor. För att identifiera vilka parametrar maskininlärningsmodellen kunde hitta mönster för valdes de parametrar med jämförbara värden, där det är möjligt att identifiera om ett värde är större eller mindre än ett annat.

Utifrån analysen valdes sex parametrar. Dessa parametrarna visas i Figur 4.1 och är markerade med grön bakgrund.

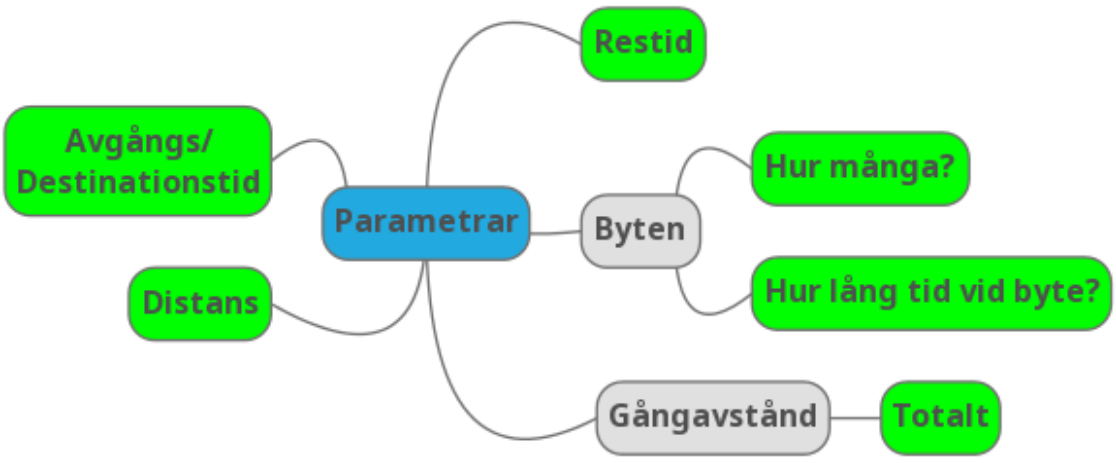


Fig. 4.1 - Parametrarna som används i maskininlärningsmodellen

4.3 Val av datafilsformat

Det initiala datasfilsformatet som togs fram bestod av åtta kolumner. Formatet visas i Figur 4.2.

```
userId,noExchanges,distance,minToDeparture,travelTime,exchangeTime,walkDist,chosen
```

Fig. 4.2 - Det initiala datafilsformatet

En resa sparas per rad och "chosen" kan anta värdena 0 eller 1 beroende på om resan valdes eller inte. "userId" visar vilken resenär resan hör till. Resterande kolumner visar hur resan rankas på dessa olika parametrar, från 0 till 10.

Efter att tester för att skapa en maskininlärningsmodell baserat på det initiala datafilsformatet inte kunde ge någon bra precision skapades ett nytt datafilsformat. För att skapa det nya datafilsformatet utfördes en analys över vad som kan ha gjort så att maskininlärningsmodellen inte kunde få en bra precision baserat på det initiala datafilsformatet. Under analysen upptäcktes det att det var svårt för maskininlärningsmodellen att hitta mönster i datan då endast en resa presenterades per rad och förutsägelserna skulle baseras på en jämförelse av resor. Därför kombinerades istället fyra resor på en rad och "chosen" visade istället vilken av resorna som valdes. En rad visas i Figur 4.3.

```
userID,chosen,
cat1,ex1,dist1,minTo1,travTime1,exTime1,walkD1,
cat2,ex2,dist2,minTo2,travTime2,exTime2,walkD2,
cat3,ex3,dist3,minTo3,travTime3,exTime3,walkD3,
cat4,ex4,dist4,minTo4,travTime4,exTime4,walkD4
```

Fig. 4.3 - Det andra datafilsformatet

Kolumnerna som avslutas med 1 representerar den första resan, de som slutar med 2 representerar den andra resan och så vidare. Värdena för kolumnerna "cat1", "cat2", "cat3" och "cat4" är alltid 1, 2, 3 och 4. Detta värde används för att kunna binda resa ett med värdet 1, resa två med värdet 2, resa tre med värdet 3 och resa fyra med värdet 4.

Precisionen för maskininlärningsmodellen när den baserades på det nya datafilsformatet blev nu bättre än precisionen för när maskininlärningsmodellen baserades på det initiala datafilsformatet, men inte tillräckligt bra. Ett tredje format togs då fram. Det tredje datafilsformatet baserades på det föregående, men delades upp så att endast två resor presenteras per rad istället för fyra. Filerna separerades även baserat på resenärer istället för att ha alla i samma fil, så kolumnen "userID" togs bort. Detta format visas i Figur 4.4.

```
chosen,
cat1,ex1,dist1,minTo1,travTime1,exTime1,walkD1,
cat2,ex2,dist2,minTo2,travTime2,exTime2,walkD2
```

Fig. 4.4 - Det tredje datafilsformatet

En av resorna är den valda resan och den andra är en ovald resa. För att spara undan data på ett sätt som maskininlärningsmodellen kan hitta mönster för skapades ett system för att tilldela parametrarna i resorna ranker baserat på deras värde i jämförelse med övriga resors parametrar. Rankning av parametrar beskrivs i kapitel 5.3.2. Alla resor rankas emot varandra innan de delas upp i grupper.

4.4 Identifiering av transformationsrecept

Vid identifiering av transformationsrecept jämfördes resultaten på de resulterande maskininlärningsmodellernas evaluering emot varandra. Vid en binär klassificeringsmodell presenteras resultatet i en graf med två kurvor. Ena kurvan visar värdena där rätt svar är 0 och den andra visar värdena där rätt svar är 1. “0-kurvan” är en grå linje med grå ifyllnad medans “1-kurvan” är en svart linje med gul ifyllnad. X-axeln i grafen visar poängen för förutsägelsen, ett högre poäng betyder mer sannolikhet att resan väljs. Y-axeln visar antalet förutsägelser som får en viss poäng. För en god precision ska dessa kurvor vara så separerade som möjligt, “0-kurvan” ska ligga åt vänster och “1-kurvan” åt höger. Det finns även ett reglage där man kan ställa in tröskelvärdet för när en förutsägelse ska räknas som 1 eller 0, förutsägelser som får en högre poäng än tröskelvärdet räknas som 1 och förutsägelser som får en lägre poäng än tröskelvärdet räknas som 0. I grafen får de förutsägelserna som förutsågs fel baserat på tröskelvärdet röda streck i sin ifyllning.

Vid en multinomial klassificeringsmodell presenteras resultatet i form av en tabell. Raderna visar de verkliga värdena och kolumnerna visar de förutsagda värdena. Diagonalen, från övre högra hörnet ner till nedre vänstra hörnet, visar korrekta förutsägelser. Dessa rutor är färgade blåa och desto mer mörkblåa rutorna är desto högre andel korrekta förutsägelser. De övriga rutorna visar felaktiga förutsägelser. Dessa rutor är färgade gula, men går mot rött desto fler felaktiga förutsägelser som maskininlärningsmodellen har gjort.

För att jämföra resultaten används maskininlärningsmodellernas precision, som definieras som antal korrekta förutsägelser dividerat med det totala antalet förutsägelser som utförts.

4.4.1 Initiala formatet

Genom att använda endast “Quantile Binning” med 5, 10 och 20 kategorier enligt transformationsreceptet som presenteras i Appendix A.1 gavs resultaten som presenteras i Figur 4.5. Linjerna överlappar i princip helt. Den svarta linjen har många värden som förutspås som “0” och inga värden som förutspås som “1”.

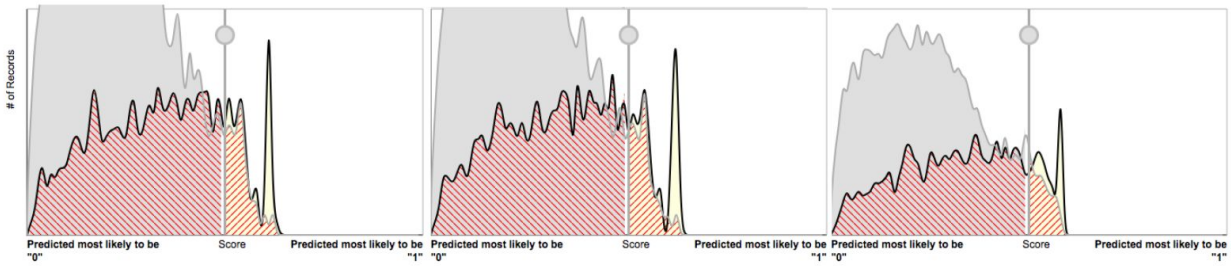


Fig. 4.5 - Visualisering av evalueringsresultaten för det initiala formatet där receptet endast består av “Quantile Binning”

För att förbättra precisionen jämfört med föregående precision lades “cartesian” till i transformationsreceptet. Med hjälp av detta binds “userID” med övriga kolumner enligt receptet som hittas i Appendix A.2. Detta gav resultatet som visas i Figur 4.6. Här sprids resultatet ut mer. Lägst till vänster finns endast den gråa linjen. Den svarta linjen har gått mer åt höger, vilket visar att maskininlärningsmodellen förutsäger fler resor med värdet “1” på “chosen” som “1”. Det finns fortfarande mycket överlappning för linjerna. Härifrån gick det inte att förbättra resultatet mer för det här formatet.

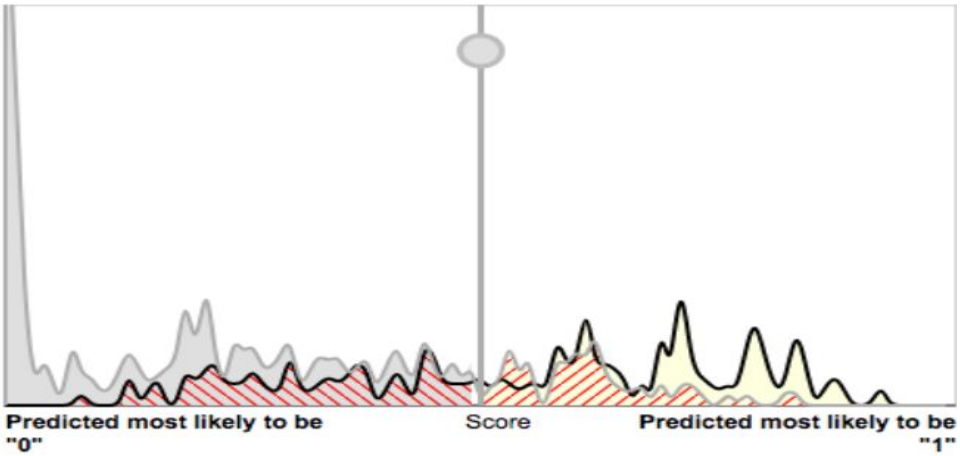


Fig. 4.6 - Visualisering av evalueringsresultaten för det initiala formatet där receptet består av “Quantile Binning” och “cartesian”

4.4.2 Andra formatet

Efter framtagandet av det andra formatet skapades ett recept som bygger på det föregående. Med "cartesian" binds "userID" och "cat" med övriga kolumner för respektive resa, enligt receptet i Appendix A.3. Resultatet presenteras i Figur 4.7. Resultatet visar att maskininlärningsmodellens förutsägelser ger en precision på 74%. Det är bättre än föregående men fortfarande inte helt pålitligt.

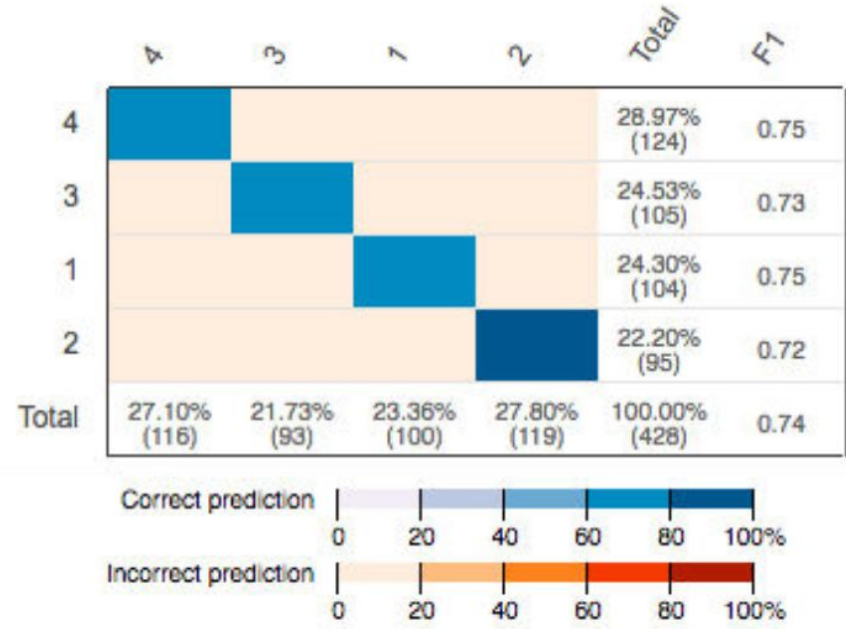


Fig. 4.7 - Visualisering av evalueringsresultaten för det andra formatet där receptet består av "Quantile Binning" och "cartesian"

För att öka precisionen testades ett recept som binder ihop kolumnerna för varje resa till varje möjlig delmängd. Då detta resulterar i totalt 256 kombinationer visas endast kombineringen av resa ett i Appendix A.4. Resultatet presenteras i Figur 4.8. Nu visar resultatet en precision på 80%. Det gick inte att förbättra precisionen mer med detta format.

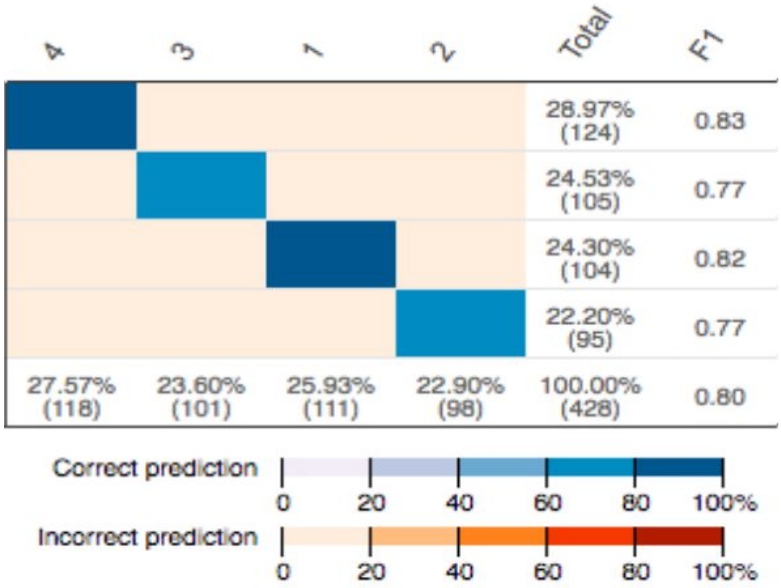


Fig. 4.8 - Visualisering av evalueringsresultaten för det andra formatet där receptet består av "Quantile Binning" och "cartesian" för alla möjliga delmängder

4.4.3 Tredje formatet

Första receptet som testades för det tredje formatet var ett recept som binder ihop alla möjliga delmängder för en resa. Kolumnerna kombineras på samma sätt som i receptet som presenteras i Appendix A.4. Resultatet presenteras i Figur 4.9. Resultatet blev en precision på 91%.

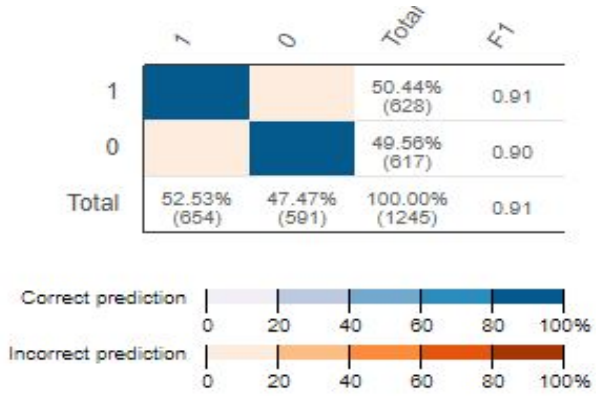


Fig. 4.9 - Visualisering av evalueringsresultaten för det tredje formatet där receptet består av "Quantile Binning" och "cartesian" för alla möjliga delmängder

Även om föregående resultat blev bra testades det om det gick att öka precisionen ytterligare och samtidigt göra receptet enklare med mindre transformeringar. Genom att binda kolumnerna mellan resorna istället för internt enligt receptet i Appendix A.5 ökade precisionen med en procent.

När man även la till en “cartesian” som binder alla kolumnerna för en resa i en lång sträng enligt receptet i Appendix A.6 steg precisionen till 98%.

5 Resultat

Resultatet från examensarbetet är en prototyp av en Android-applikation med tillhörande maskininlärningsmodeller samt en utvärdering av maskininlärningsmodellerna. I kapitlet kommer slutresultatet av examensarbetet presenteras.

5.1 Översiktlig bild av Android-prototypen

En del av examensarbetets resultat är en Android-prototyp där en resenär kan söka efter och välja resor mellan två platser. Vid sökning efter resor hämtas reseinformation från Google Directions API. När en resa valts sparas datan för resan undan i Amazon S3 och används sedan i Amazon Machine Learning för att förutsäga vilka resor resenären är mest sannolik att välja vid nästa sökning. I Figur 5.1 presenteras ett sekvensdiagram som visar vad som händer när en resenär söker efter resor, klickar in på en specifik resa och sedan väljer resan.

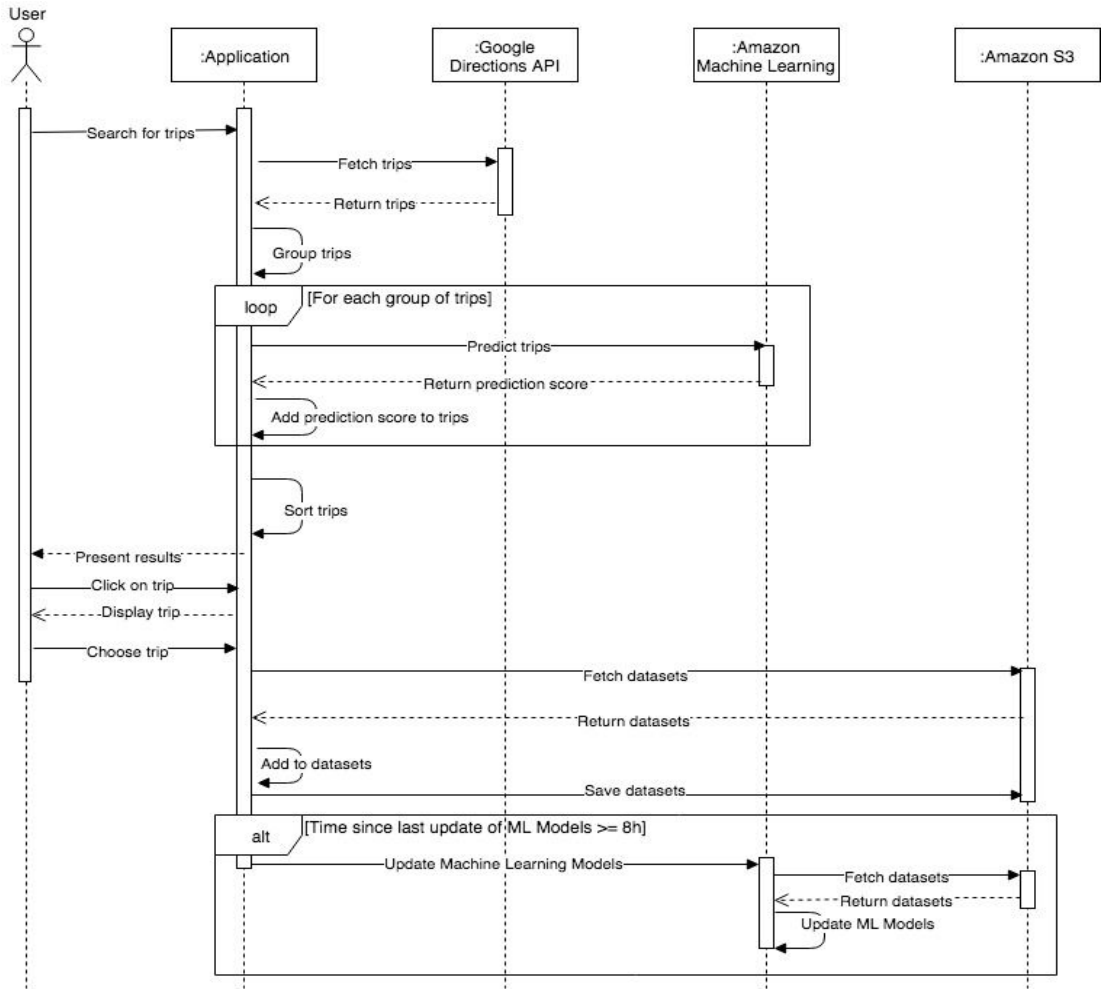


Fig. 5.1 - Sekvensdiagram som beskriver vad som händer när en resenär söker efter resor, klickar in på en resa och sedan väljer resan.

5.2 Maskininlärningsmodeller

För en resenär kombineras tre maskininlärningsmodeller. Anledningen till att tre maskininlärningsmodeller används är för att få ut bättre resultat genom att kombinera maskininlärningsmodeller som behandlar olika typer av dataset. Maskininlärningsmodellerna använder sig i övrigt av samma recept och inställningar. Följande maskininlärningsmodeller används:

- **Global maskininlärningsmodell:** Maskininlärningsmodell som baseras på ett dataset som består av data för alla resenärers resor.
- **Stor maskininlärningsmodell:** Maskininlärningsmodell som baseras på ett dataset som består av data för en enskild resenärs alla resor.
- **Liten maskininlärningsmodell:** Maskininlärningsmodell som baseras på ett dataset som består av data för en enskild resenärs 100 senaste datarader (cirka 15 resor).

Maskininlärningsmodellerna har olika funktioner och påverkar resvägsförlagens sortering med olika vikter. Vikterna baseras på hur stor andel av det slutgiltiga resultatet från maskininlärningsmodellerna den individuella maskininlärningsmodellen ska påverka. Vikterna har bestämts baserat på vilken funktion maskininlärningsmodellen ska uppfylla, vilket beskrivs i kapitel 5.2.1, 5.2.2 och 5.2.3.

5.2.1 Liten maskininlärningsmodell

Den lilla maskininlärningsmodellen har störst påverkan på sorteringen och står för 65% av det slutliga resultatet. Funktionen för den här maskininlärningsmodellen är att snabbt reagera på förändring i en resenärs preferenser. Genom att hålla datasetet litet kommer förändringar på en resenärs preferenser få en större påverkan för vilken resa maskininlärningsmodellen anser vara den bästa matchningen. Detta då datasetet alltid består av cirka 15 resor så varje individuell resa kommer stå för cirka 6,7% av det totala datasetet. Vid åtta resor med nya preferenser kommer den typen av resor vara dominerande i datasetet och därmed kommer maskininlärningsmodellen ge förutsägelser som främst baseras på de nya preferenserna.

5.2.2 Stor maskininlärningsmodell

Den stora maskininlärningsmodellen har mindre påverkan på sorteringen och står för 25% av det slutliga resultatet. Funktionen med maskininlärningsmodellen är att ge en grund för vilka typer av resor en resenär föredrar. Då en resenärs alla resor sparas kommer maskininlärningsmodellen ge en förutsägelse baserat på de preferenser resenären har haft under längst tid. Genom att kombinera resultaten från den stora och den lilla maskininlärningsmodellen i prototypen kommer inte en resenärs små avsteg från preferenserna ha någon större effekt på hur datan sorteras.

5.2.3 Global maskininlärningsmodell

Den globala maskininlärningsmodellen har minst påverkan på sorteringen och står för 10% av det slutliga resultatet. Syftet med den här maskininlärningsmodellen är att ge en överblick över hur resvanorna för alla resenärer ser ut. Genom att endast tilldela den globala maskininlärningsmodellen 10% av det slutgiltiga resultatet påverkas inte den individuella resenärens sökresultat på ett sådant sätt att resultatet blir felaktigt om resenärens preferenser inte stämmer överens med preferenserna i det globala datasetet. Istället ska maskininlärningsmodellen skilja på två resor som förutses vara ungefär lika bra matchningar enligt resenärens stora och lilla maskininlärningsmodell, det vill säga att de har blivit tilldelade ungefär lika många poäng. Om en resenär inte har några sparade resor kommer den globala maskininlärningsmodellen stå för hela det slutliga resultatet då resenären inte har några övriga maskininlärningsmodeller ännu.

5.2.4 Parametrar som används i maskininlärningsmodellen

Efter att möjliga parametrar diskuterats fram mellan examensarbetaren och handledaren på företaget och en analys för att få bort orimliga och irrelevanta parametrar var utförd av examensarbetaren kvarstod parametrarna som hittas i Appendix B.1. Dessa parametrar kan alla användas för att göra så att resförslagen för en resenär matchar resenärens personliga preferenser.

Efter en djupare analys på parametrarna, som beskrivs i kapitel 4.2, valdes sex parametrar som maskininlärningsmodellen ska använda. De valda parametrarna visas i Figur 5.2 och är markerade med grön bakgrund.

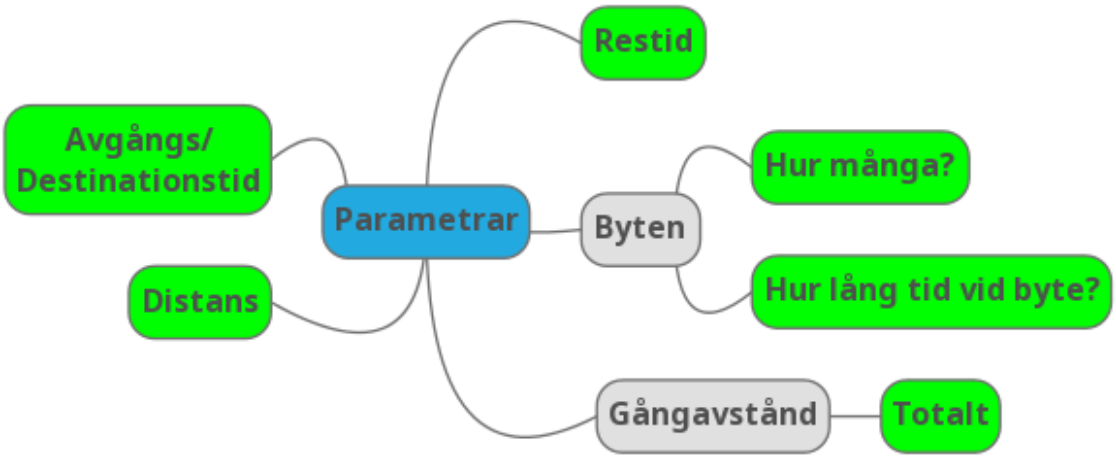


Fig. 5.2 - Parametrarna som används i maskininlärningsmodellen

Parametrarna ger en överblick över vilka typer av resor en resenär väljer när resenären reser.

5.2.5 Transformationsrecept

Transformationsreceptet för parametrarna som används i maskininlärningsmodellerna finns i Appendix B.2. Receptet resulterar i tre outputs; de otransformerade kolumnerna, en sträng som består av en resas alla kolumner bundna med hjälp av “cartesian” och strängar som består av bindningar mellan de två resornas kolumner.

De otransformerade kolumnerna används som en utgångspunkt för att maskininlärningsmodellen ska kunna hitta mönster. Genom att använda en sträng med en resas alla kolumner bundna kan maskininlärningsmodellen hitta mönster för hur den valda resans parametrar oftast ser ut. Strängarna som består av bindningar mellan de två resornas kolumner ger maskininlärningsmodellen möjligheten att hitta mönster baserat på hur den valda resans parametrar är i relation till övriga resor, såsom om den valda resan har ett lägre antal byten eller kortare restid jämfört med de övriga resorna.

5.3 Android-prototyp

Prototypen av Android-applikationen utvecklades för att användas som ett exempel på hur maskininlärningsmodellen kan användas i praktiken. I det här kapitlet ges fakta över hur applikationen använder och samspelar med maskininlärningsmodellerna samt tillhörande funktioner.

5.3.1 Översiktlig bild av prototypen

Prototypen erbjuder funktionalitet för att söka efter resförslag, se information om resförslagen och välja en resa. Figur 5.3 illustrerar hur resultatet vid sökning efter resförslag presenteras. De tre resförslag som maskininlärningsmodellerna anser vara bäst matchande med resenärens preferenser presenteras högst upp i listan med en gul bakgrundsfärg. Resterande resor presenteras sorterade efter ankomsttid.

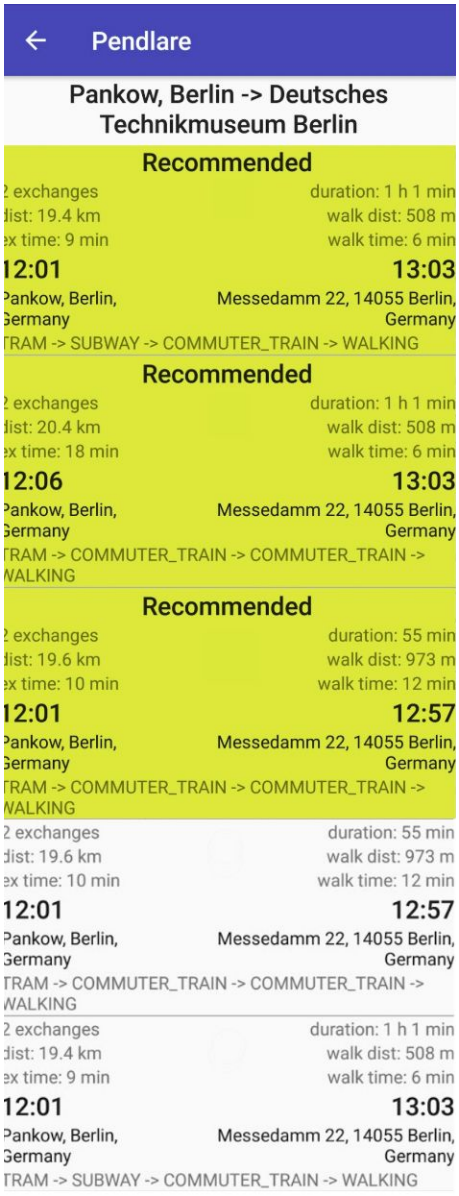


Fig. 5.3 - Presentation av resvägsförslag i prototypen

Figur 5.4 illustrerar hur informationen om en specifik resa presenteras i prototypen samt var resenären kan välja resan.

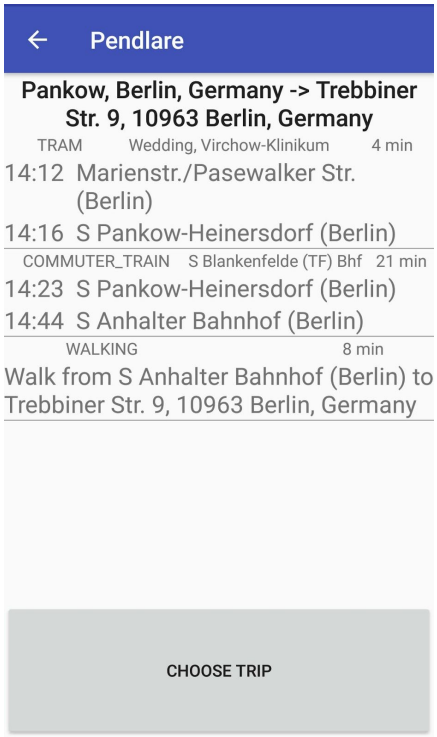


Fig. 5.4 - Gränssnitt för information om en specifik resa samt för att välja en resa

De fullständiga gränssnitten för prototypen presenteras i Appendix C.2.

5.3.2 Rankning av resvägsdata

För att kunna jämföra resor i maskininlärningsmodellen krävs ett system för att ranka resornas parametrar emot varandra. Rankning av parametrar består av tre steg:

1. **Skapa gränser för rankningar:** Här beräknas medelvärdet för varje parameter mellan resorna. Utifrån detta värde skapas 11 grupper (0-10) per parameter som beskriver vilka värden som får vilken rank. Gränserna för varje rankning delas upp enligt följande: Medelvärdet divideras med 5. Kvoten ur divisionen benämns k för enklare referens. Rank 0 innehåller värden som är lägre än k. Rank 1 innehåller värden som är högre eller lika med k, men lägre än 2*k. Rank 2 innehåller värden som är högre eller lika med 2*k, men lägre än 3*k. Detta gör man fram till rank 10 där alla parametrar med ett värde som är lika med eller högre än 10*k placeras.
2. **Ranka resorna:** Rankning sker genom att utifrån gränserna tilldela parametrarna i en resa en rank och placera dessa i en mapp per resa där namnet på parametrarna är nycklar och motsvarande ranker värden.
3. **Normalisera rankerna:** För att maskininlärningsmodellen ska få konsistent data att jobba med normaliseras rankerna så att lägsta ranken för varje parametergrupp alltid har

värdet 0. Detta sker genom att i varje parametergrupp hitta lägsta värdet och sedan subtrahera detta från alla ranker i gruppen.

5.3.3 Lagring av data

Varje gång en resenär väljer en resa sparas resdatan i Amazon S3. Processen för att spara resdatan är följande:

1. Resenären väljer en resa.
2. Alla resor som presenterats för resenären rankas enligt rankningssystemet som beskrivs i kapitel 5.3.2.
3. För varje resa som inte valdes skapas en sträng innehållande den valda resans rankningar samt den ej valda resans rankningar.
4. Användarens filer läses in från Amazon S3.
5. De nya strängarna läggs till i filerna.
    a. Om summan av antalet i rader i det lilla datasetet adderat med antalet rader som ska läggas till är större än 100 tas de tidigast inlagda raderna bort från datasetet tills summan blir lika med 100 (first-in, first-out).
6. Filerna läggs in i Amazon S3 igen.

5.3.4 Användning av Amazon Machine Learning för sortering av resvägsresultat

Vid sortering av resorna jämförs de individuella resornas poäng som tilldelas från maskininlärningsmodellerna. Poängen motsvarar sannolikheten att resan blir vald. Då maskininlärningsmodellerna tar två resor som indata och en sökning efter resor presenterar fler resor än det behöver resultaten delas in i grupper. För att kunna göra en rättvis bedömning mellan resorna grupperas alla resor med varandra, vilket resulterar i $\frac{n!}{2(n-2)!}$ grupper, där n är antalet resor sökningen resulterade i. För två resor blir det en grupp, för tre resor blir det två grupper, för fyra resor blir det sex grupper och så vidare.

För varje grupp utförs sedan tre förutsägelser, en för varje maskininlärningsmodell, och de resulterande poängen multipliceras med vikten för motsvarande maskininlärningsmodell för att sedan adderas till resornas totala maskininlärningspoäng. Processen för att skapa en förutsägelse för två resor, resa 1 och resa 2, illustreras i Figur 5.5.

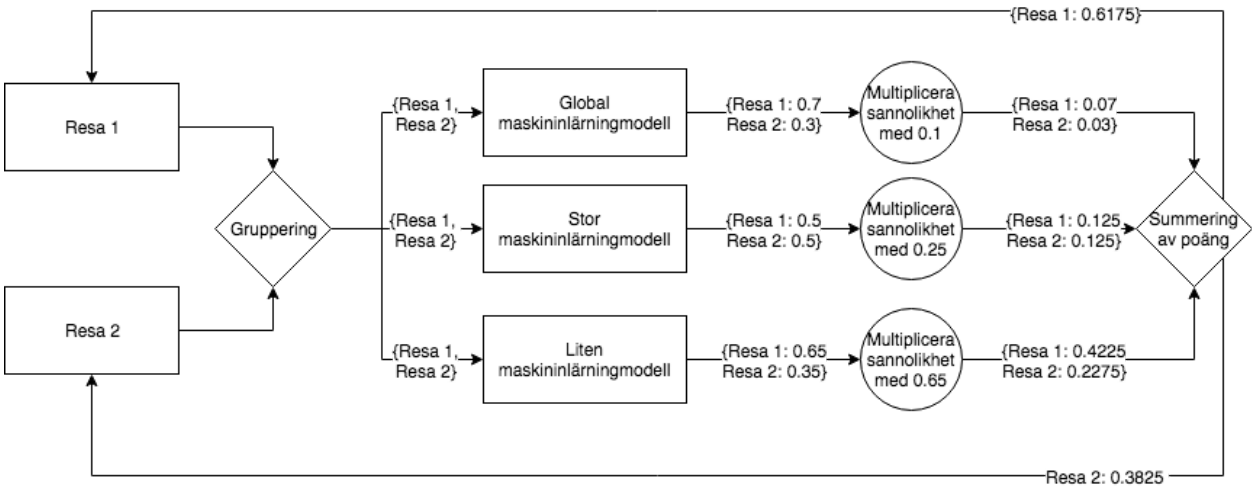


Fig. 5.5 - Illustration över hur processen för att skapa en förutsägelse för två resor ser ut

Utifrån maskininlärningspoängen sorteras sedan resorna och de tre översta kopieras till en separat lista med oförändrad inbördes ordning. Originallistan med resorna sorteras sedan utifrån ankomsttid. När resultaten sedan presenteras visas först listan med resor sorterade efter maskininlärningspoäng, med gul bakgrundsfärg för att de ska sticka ut, och därefter originallistan.

5.3.5 Uppdatering av maskininlärningsmodeller

Maskininlärningsmodellerna uppdateras automatiskt i bakgrunden, även när resenären inte har prototypen igång. Uppdateringarna är schemalagda till att ske var åttonde timme, och endast om en resenär har någon ny data i sina dataset. Anledningen till att uppdateringarna sker var åttonde timme är för att detta resulterar i tre uppdateringar per dag. Det är tillräckligt ofta för att ny data snabbt ska inkluderas i maskininlärningsmodellerna samtidigt som det inte kräver mycket resurser för att hålla maskininlärningsmodellerna uppdaterade.

5.4 Testresultat

Testresultaten visar att maskininlärningsmodellerna ger pålitliga förutsägelser med en bra precision. Nedan presenteras en liten del av testresultaten. De fullständiga testresultaten presenteras i Appendix B.3.

I Tabell 5.1 visas testresultaten för maskininlärningsmodellernas precision för olika resenärer med olika preferenser. Ur testresultaten går det att se att precisionen för maskininlärningsmodellerna i genomsnitt ligger över 90%. Detta visar på att resultaten från maskininlärningsmodellerna är pålitliga och ger förutsägelser som stämmer överens med verkligheten.

Tabell 5.1 - Tabell som visar precisionen för maskininlärningsmodellerna för olika resenärer

Antal rader i träningsdatan	Precision för resenär 1	Precision för resenär 2	Precision för resenär 3	Precision för resenär 4	Genomsnittlig precision
1000	96,4%	98,9%	97,9%	97,4%	98,0%
400	94,8%	97,1%	96,8%	96,1%	96,2%
100	85,3%	93,6%	91,8%	93,2%	91,0%

I Figur 5.6 visas sökresultaten för tre resenärer med olika preferenser. Den första resenärens preferenser är kort tid vid byten och kort tid till avgång, den andra resenärens preferenser är kort tid till avgång och kort gångavstånd och den tredje resenärens preferenser är kort resedistans och kort tid till avgång.

Fig. 5.6 - Sökresultat för tre resenärer med olika preferenser

Ur Figur 5.6 kan man se att de rekommenderade resorna är olika baserat på resenärens preferenser samt att de är sorterade utifrån resenärernas preferenser. Detta visar att maskininlärningsmodellen gör korrekta förutsägelser baserade på resenärens preferenser.

Ett test utfördes för att se om maskininlärningsmodellerna reagerar på förändringar i preferenser. För att utföra testet användes två dataset med 100 rader i varje dataset. Dataset 1 innehåller resor med följande preferenser:

1. Lågt antal byten

2. Kort tid till avgång

och dataset 2 innehåller resor med följande preferenser:

1. Kort tid till avgång
2. Gångavstånd

Under testets gång byttes raderna i dataset 1 successivt ut mot raderna i dataset 2 med 20 rader per gång. Resultatet för testerna presenteras i Figur 5.7 i form av sex bilder där varje bild visar sökresultaten för resenären under tiden preferenserna bytts ut.

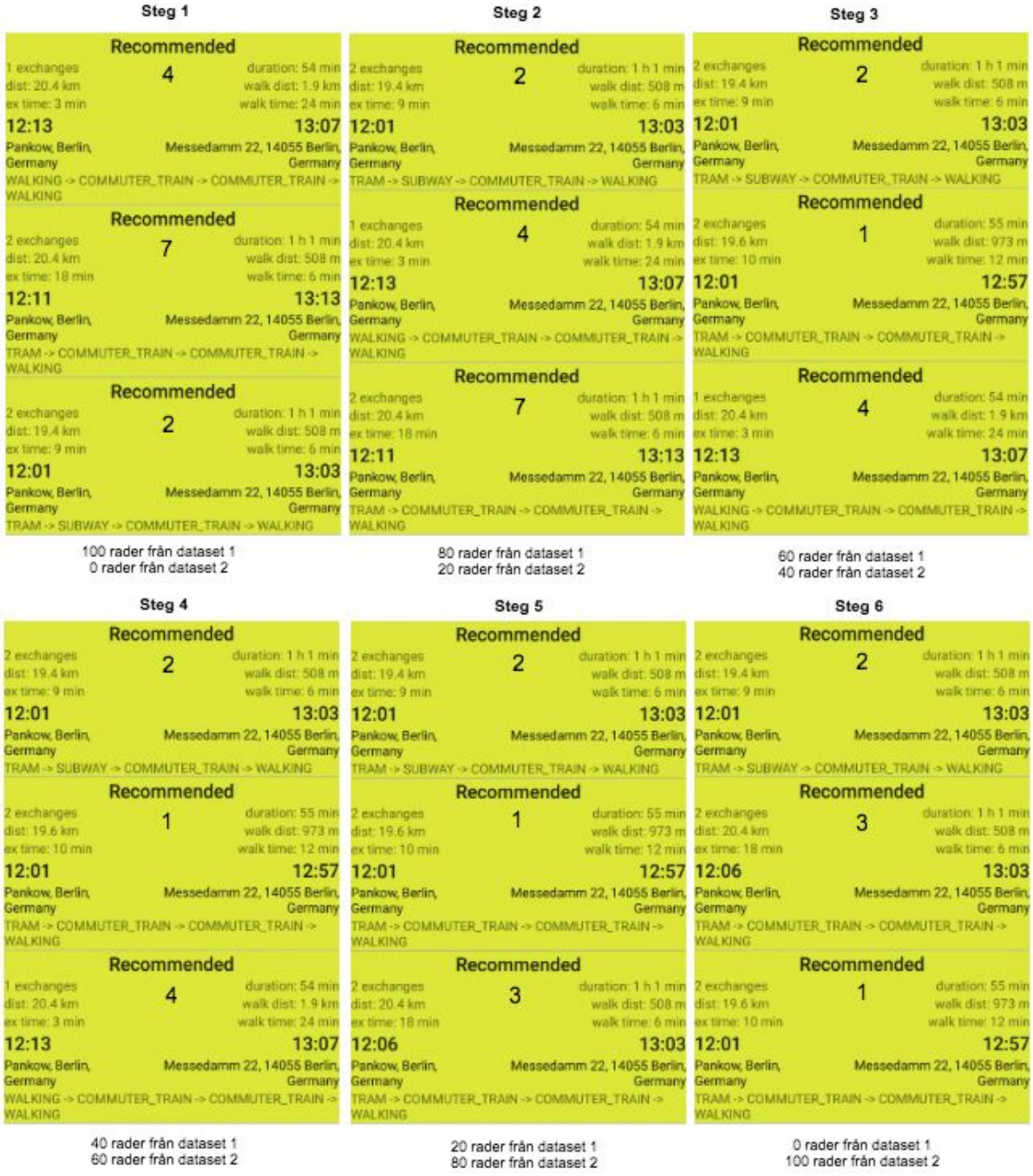


Fig. 5.7 - Sökresultaten för en resenär som successivt byter ut sina preferenser.

Ur Figur 5.7 kan man se att sökresultaten påverkas redan i steg två där 20 rader från dataset 1 har ersatts med 20 rader från dataset 2. Resa 2 har flyttats upp till plats ett, då den har kortast tid till avgång och lägst gångavstånd vilket stämmer överens med preferenserna i dataset 2. Preferenserna i dataset 1 är fortfarande de som påverkar mest.

I steg 3 och steg 4, där 40 respektive 60 rader från dataset 1 har ersatts med 40 respektive 60 rader från dataset 2, försvinner resa 7 och ersätts med resa 1. Resa 1 är den resa med kortast tid till avgång, vilket stämmer överens med preferenserna i dataset 2. Resa 4 flyttas ner till plats tre då den har längre tid till avgång och längre gångavstånd än resa 1 och resa 2. Preferenserna i dataset 1 påverkar fortfarande resultatet vilket man kan se på att resa 4 fortfarande är kvar, fastän den har lång tid till avgång och långt gångavstånd.

I steg 5, där 80 rader från dataset 1 har ersatts med 80 rader från dataset 2, försvinner resa 4 och ersätts med resa 3. Då resa 3 både har kortare tid till avgång och lägre gångavstånd men ett högre antal byten än resa 4 visar detta att preferenserna från dataset 2 har tagit över och maskininlärningsmodellen nu ger förutsägelser baserat på de nya preferenserna. I steg 6 försvinner inga resor utan sorteras istället internt efter deras gångavstånd, enligt preferens 2 i dataset 2.

Utifrån resultatet i det här testet kan en slutsats dras om att maskininlärningsmodellerna snabbt reagerar på förändringar i preferenser hos en resenär. Redan efter 20 ersatta rader (cirka 3 resor) har preferenserna börjar påverka resultatet. Efter 80 ersatta rader (cirka 11-12 resor) finns det endast en liten påverkan från de gamla preferenserna och efter 100 ersatta rader (cirka 15 resor) har de nya preferenserna tagit över.

6 Slutsats

Det är fullt möjligt att använda maskininlärning för att förbättra hur resvägsförslagen presenteras i en reseapplikation. Viktigt att notera är att maskininlärningsmodellen måste ha hög träffsäkerhet och snabbt kunna reagera på förändringar för att inte ge resenären irrelevanta resvägsförslag vilket ger en sämre användarupplevelse.

Genom att använda Amazon Machine Learning kan man få fram maskininlärningsmodeller som ger god precision baserat på parametrarna som har används i det här examensarbetet. Testerna av maskininlärningsmodellerna har visat på att maskininlärningsmodellerna reagerar på förändring av preferenser, vilket visas i Appendix B.3.6, och ger relevanta resvägsförslag baserat på en resenärs tidigare resor, vilket visas i Appendix B.3.

Följande frågor besvaras av examensarbetet:

1. Vilka parametrar finns det som kan påverka en resenärs val av resväg?
2. Vilket eller vilka transformationsrecept ska användas i maskininlärningsmodellen för att önskat resultat ska uppnås?
3. Hur ska man gå tillväga för att, utifrån data från maskininlärningsmodellen, sortera resultaten efter relevans?

Svaren på ovanstående frågor presenteras nedan.

Av examensarbetet framgår det att det fanns ett antal parametrar som påverkar en resenärs val av resväg. Parametrarna som tagit fram under examensarbetet hittas i Appendix B.1. Parametrarna kan delas in i två typer av parametrar, direkta och indirekta.

Transformationsreceptet för maskininlärningsmodellen som används i examensarbetet hittas i Appendix B.2. Receptet baseras på att två resor jämförs med varandra där en av resorna har större sannolikhet att väljas av resenären än den andra. Receptet transformerar parametrarna på två olika sätt. Parametrarna för resorna korsas mellan resorna för att visa mönster över vilka parametrar en resenär föredrar. Parametrarna knyts även ihop med “Cartesian Product” inom en resa för att visa mönster för de enskilda resorna. Recept är en viktigt del i Amazon Machine Learning och är något som behöver anpassas baserat på formatet på datasetet. När man ska skapa en maskininlärningsmodell är receptet något man måste spendera tid på att utforma för att få så bra precision som möjligt.

När ett recept är framställt och maskininlärningsmodellen är aktiv måste ett system för att använda resultaten i praktiken finnas. Under examensarbetets gång har det framkommit att bäst resultat ges om man kombinerar resultaten från tre maskininlärningsmodeller, som beskrivs i

kapitel 5.2. Sortering av resorna sker genom att man tilldelar varje resa en poäng som motsvarar sannolikheten att en viss resa väljs. För att poängen ska ge rätt resultat anropas maskininlärningsmodellerna för varje par av resor som går att skapa ifrån listan på resor och resultatet ifrån maskininlärningsmodellerna summeras för varje resa. En slutsats drogs även utifrån en diskussion mellan examensarbetaren och handledaren på företaget över att det bästa tillvägagångssättet för att presentera resorna var att endast presentera de tre bäst matchade resorna sorterade efter maskininlärningspoängen och sedan sortera listan utifrån ankomsttid. Den här presentationstekniken var fördelaktigt då resenären inte alltid utgår från sina normala preferenser och därför är det bra med en lista där resultaten är sorterade utifrån en gemensam parameter.

6.1 Reflektion över etiska aspekter

För att använda en maskininlärningsmodell behöver man data. Insamling av data från användare är ett ämne som ofta är uppe för debatter när det handlar om applikationer och hemsidor då användarna kan anse att detta kränker deras integritet. Därför är det viktigt att man endast samlar den data man behöver och säkerställer att en användares data inte direkt kan knytas till användaren. I examensarbetet löses detta genom att endast spara undan den data som används i maskininlärningsmodellerna och att all data sparas under ett användar-id. Detta id kan inte kopplas till någon resenär utan resenärens enhet är det enda som håller reda på detta. Genom att använda detta tillvägagångssättet försäkrar man att inga personliga uppgifter sparas på något sätt.

Syftet med examensarbetet är att i förlängningen få fler människor att välja kollektivtrafiken med hjälp av maskininlärning. Detta leder till mindre biltrafik som i sin tur leder till mindre klimatpåverkan. Då miljön på jorden bara blir sämre är det viktigt att försöka påverka människor att välja klimatsmarta alternativ, och det här examensarbetet är ett steg för att uppnå detta.

6.2 Framtida utvecklingsmöjligheter

Resultatet från examensarbetet går att utveckla och förbättra på flera sätt. I det här kapitlet ges några exempel över vilka förbättringar som kan utföras.

6.2.1 Lägga till parametrar

För att förbättra effekten från maskininlärningen kan man lägga till fler parametrar. Genom att lägga till parametrar kan maskininlärningsmodellerna förutsäga sannolikheten att en resenär väljer en viss resa baserat på fler parametrar. Man får då maskininlärningsmodeller som är anpassade efter fler resenärer då olika resenärers preferenser kan skilja sig åt på olika sätt. Maskininlärningsmodellerna kan även hitta komplexare relationer mellan preferenserna hos en resenär om antalet parametrar ökar. Ett antal parametrar som kan användas är de som visas i

Appendix B.1. Bland dessa finns det två kategorier: Parametrar som hämtas från information om en resa och parametrar som hämtas från utomstående källor. Antal stopp, färdtyp och huruvida resan är handikappanpassad eller inte är exempel på parametrar som hämtas från informationen om en resa. Väder och trafik är exempel på parametrar som hämtas från utomstående källor.

För att lägga till parametrar som hämtas från informationen om en resa ändras formatet på dataseten för att möjliggöra användning av de nya parametrarna. Viktigt att notera är att inte lägga till för många parametrar då detta kan ha en negativ påverkan på precisionen hos maskininlärningsmodellerna. Det blir svårare för maskininlärningsmodellerna att hitta mönster i ett dataset om allt för många parametrar finns med då komplexiteten för relationerna mellan parametrarna ökar. Om komplexiteten blir för hög kommer maskininlärningsmodellerna ha svårt att fastställa matematiska funktioner som ger korrekta resultat.

För att använda utomstående parametrar kan två metoder tillämpas. Parametrarna kan läggas till i dataseten precis som för parametrar från resor, eller kan de användas som en nyckel över vilken maskininlärningsmodell som ska användas. Om parametern används som nyckel kontrolleras först värdet på parametern när en sökning utförs. Utifrån värdet på parametern väljs vilken maskininlärningsmodell som ska användas. Det är då även fördelaktigt att ha en maskininlärningsmodell som använder resenärens samtliga resor för att undvika felaktiga resultat för de maskininlärningsmodeller som används mindre ofta.

6.2.2 Använda Amazon Sagemaker

När maskininlärningsmodellen ska utvecklas och fler parametrar ska användas är det fördelaktigt att använda en mer flexibel tjänst för att skapa maskininlärningsmodeller. Amazon Sagemaker är ett exempel på en sådan tjänst. Genom Amazon Sagemaker kan maskininlärningsmodellen exempelvis hantera val av maskininlärningsmodell baserat på en parameter som nyckel och kombinera den lilla, stora och globala maskininlärningsmodellen. Man kan därmed lägga över ansvaret som har med maskininlärning att göra till maskininlärningsmodellen istället för att föra det i Android Studio.

Det är även möjligt att välja bland femton maskininlärningsalgoritmer, till skillnad från Amazon Machine Learnings tre maskininlärningsalgoritmer, och man kan även använda egenskrivna maskininlärningsalgoritmer (Amazon Web Services, u.åb). Detta leder till en maskininlärningsmodell som bättre kan anpassas utefter vilket resultat som önskas.

6.2.3 Användartestning

Det skulle vara fördelaktigt om ett användartest utfördes på prototypen för att utvärdera maskininlärningsmodellerna. Ett antal resenärer som regelbundet pendlar med kollektivtrafiken

skulle kunnat få använda Android-prototypen. Varje gång resenären utför en resa väljer de även resan i Android-prototypen, som sparas undan i AWS S3 och används i maskininlärningsmodellerna. Efter ett antal dagar börjar resenärerna anteckna vilka resor som rekommenderas och huruvida dessa stämmer överens med deras valda resa eller inte. Resenären antecknar även vilken resa resenären väljer istället och varför den föredrog den resan framför de rekommenderade resorna. På så sätt kan en bild av hur maskininlärningsmodellernas resultat stämmer överens med resenärens verkliga resebeteende samt om det är några parametrar som har missats men som är viktiga för resenärerna.

Att ha ett användartest skulle ge möjlighet att stämma av så att maskininlärningsmodellerna verkligen ger önskad precision och därmed fungerar som de ska.

6.2.4 Datainsamling

Under informationsinsamlingsfasen, steg 8, gjordes ett försök att hitta ett publikt dataset som innehöll information om resenärers resvanor. Då inget sådant dataset hittades genererades egen data. Detta medförde att datan inte var en riktig representation av resenärernas resvanor, och gav inte en verklig bild över hur resenärerna använder kollektivtrafiken.

Ett tillvägagångssätt för att samla in verklig resdata skulle varit att skapa en applikation som samlar in sådan data från en grupp resenärer. Ett sådant projekt sker just nu i Berlin. Projektet heter BVG-Motion och är ett samarbete mellan BVG och MotionTag (MotionTag, u.å.). Där erbjuds resenärer att installera en applikation på sin mobiltelefon. Applikationen samlar information om resenärens position, resenärens rörelsemönster, vädret, tidpunkt på dygnet med mera. Utifrån informationen kartläggs resenärens resmönster, vilka färdtyper som föredras, vilken tidpunkt på dygnet flest resenärer åker kollektivt, vilka platser där flest resenärer hoppar på/hoppar av med mera.

Utifrån den insamlade datan skulle en analys utförts för att hitta de parametrar som är viktiga för resenärerna och därefter skapa ett datafilsformat för att skapa ett träningsset och ett evalueringsset utifrån den insamlade datan. Maskininlärningsmodellen skulle då kunna testas mot verklig data och utvecklas för att fungera optimalt för en verklig resenär.

7 Terminologi

AWS - Amazon Web Services, en samling molntjänster utvecklade av Amazon.

Amazon S3 - Amazon Simple Storage Service, en lagringstjänst som tillhandahålls från *AWS*.

Amazon SageMaker - En tjänst från AWS som möjliggör skapandet av maskininlärningsmodeller. Till skillnad från Amazon Machine Learning har användaren större kontroll över maskininlärningsprocessen.

Google Directions API - Ett API utvecklad av Google som tillhandahåller information om resvägar mellan två punkter.

JSON - JavaScript Object Notation, ett textbaserat filformat som ofta används vid utbyte av data mellan server och applikation.

Precision - Ett värde för att mäta hur en maskininlärningsmodell presterar. Definieras som antalet korrekta förutsägelser dividerat med det totala antalet förutsägelser.

Preferenser - Parametrar ur resorna som en resenär föredrar på ett visst sätt, till exempel kort tid till avgång eller få byten.

Resenär - Användare av tjänsten, någon som reser med kollektivtrafiken.

Transformationsrecept - Recept på hur kolumnerna i Amazon Machine Learning ska bearbetas innan de används som träningsdata.

8 Källor

Amazon, u.å. Global Cloud Infrastructure [Online]
Tillgänglig: https://aws.amazon.com/about-aws/global-infrastructure/
[2018-10-31]

Amazon Web Services, u.åa, Amazon Machine Learning [Online]
Tillgänglig:
https://docs.aws.amazon.com/machine-learning/latest/dg/what-is-amazon-machine-learning.html
[2018-11-14]

Amazon Web Services, u.åb, Amazon Sagemaker [Online]
Tillgänglig:
https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html
[2018-11-14]

BVG-Motion, u.å, MotionTag [Online]
Tillgänglig:
https://bvg.motion-tag.com
[2018-11-29]

D. McKenna, 2016. The Art of Scrum. 1 uppl. Aliquippa, Pennsylvania: Apress

E. Alpaydin, 2010. Introduction to Machine Learning. 2 uppl. Cambridge: The MIT Press

Google, u.å. Gson User Guide [Online]
Tillgängligt: https://github.com/google/gson/blob/master/UserGuide.md
[2019-02-05]

Miroslav Kubat, 2017. An Introduction to Machine Learning. 2 uppl. New York: Springer

ReactiveX, u.å. RxJava [Online]
Tillgänglig: https://github.com/ReactiveX/RxJava
[2018-10-31]

Square, u.å. Retrofit2 [Online]
Tillgänglig: https://square.github.io/retrofit/
[2018-10-31]

Udacity, u.å. Android Developer Guide [Online]
Tillgänglig: https://www.udacity.com/sitemap/guides/androiddeveloper
[2018-10-01]

Yufeng Guo, 2017. The 7 Steps of Machine Learning. Towards Data Science. 31 Augusti.
Tillgänglig: https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e
[2018-11-23]

Appendix A - Identifiering av recept

Appendix A.1 - Endast “Quantile Binning”

```
{
  "groups": {
  },
  "assignments": {
   "binned_walk" : "quantile_bin('Walking Distance',10)",
   "binned_exT" : "quantile_bin('Exchange Time',10)",
   "binned_travT" : "quantile_bin('Travel Time',10)",
   "binned_minTo" : "quantile_bin('Minutes To Departure',10)",
   "binned_dist" : "quantile_bin('Distance',10)",
   "binned_exc" : "quantile_bin('Exchanges',10)"
   },
  "outputs": [
    "ALL_NUMERIC",
    "ALL_CATEGORICAL",
    "binned_walk",
    "binned_exT",
    "binned_travT",
    "binned_minTo",
    "binned_dist",
    "binned_exc"
  ]
}
```

Appendix A.2 - “Quantile Binning” och “Cartesian”

```
{
  "groups": {
    "NUMERIC_VARS_QB_20": "group('Minutes To Departure','Travel
Time','Exchange Time','Exchanges','Distance','Walking Distance')"
  },
  "assignments": {
   "binned_walk" : "quantile_bin('Walking Distance',100)",
   "binned_exT" : "quantile_bin('Exchange Time',100)",
   "binned_travT" : "quantile_bin('Travel Time',100)",
```

```
   "binned_minTo" : "quantile_bin('Minutes To Departure',100)",
   "binned_dist" : "quantile_bin('Distance',100)",
   "binned_exc" : "quantile_bin('Exchanges',100)"
   },
  "outputs": [
    "ALL_BINARY",
    "ALL_CATEGORICAL",
    "quantile_bin(NUMERIC_VARS_QB_20,200)",
    "cartesian('userID', binned_walk)",
    "cartesian('userID', binned_exT)",
    "cartesian('userID', binned_travT)",
    "cartesian('userID', binned_minTo)",
    "cartesian('userID', binned_dist)",
    "cartesian('userID', binned_exc)"
  ]
}
```

Appendix A.3 - “Quantile Binning” och “Cartesian” för andra formatet

```
{
  "groups": {
    "GROUP_1": "group('ex1', 'dist1', 'minTo1', 'travTime1',
'exTime1', 'walkD1')",
    "GROUP_2": "group('ex2', 'dist2', 'minTo2', 'travTime2',
'exTime2', 'walkD2')",
    "GROUP_3": "group('ex3', 'dist3', 'minTo3', 'travTime3',
'exTime3', 'walkD3')",
    "GROUP_4": "group('ex4', 'dist4', 'minTo4', 'travTime4',
'exTime4', 'walkD4')"
  },
  "assignments": {
    "binned_1" : "quantile_bin(GROUP_1, 50)",
    "binned_2" : "quantile_bin(GROUP_2, 50)",
    "binned_3" : "quantile_bin(GROUP_3, 50)",
    "binned_4" : "quantile_bin(GROUP_4, 50)",
    "bin_1.1" : "quantile_bin(ex1, 10)",
    "bin_1.2" : "quantile_bin(dist1, 10)",
    "bin_1.3" : "quantile_bin(minTo1, 10)",
    "bin_1.4" : "quantile_bin(travTime1, 10)",
    "bin_1.5" : "quantile_bin(exTime1, 10)",
    "bin_1.6" : "quantile_bin(walkD1, 10)",
```

```
    "bin_2.1" : "quantile_bin(ex2, 10)",
    "bin_2.2" : "quantile_bin(dist2, 10)",
    "bin_2.3" : "quantile_bin(minTo2, 10)",
    "bin_2.4" : "quantile_bin(travTime2, 10)",
    "bin_2.5" : "quantile_bin(exTime2, 10)",
    "bin_2.6" : "quantile_bin(walkD2, 10)",
    "bin_3.1" : "quantile_bin(ex3, 10)",
    "bin_3.2" : "quantile_bin(dist3, 10)",
    "bin_3.3" : "quantile_bin(minTo3, 10)",
    "bin_3.4" : "quantile_bin(travTime3, 10)",
    "bin_3.5" : "quantile_bin(exTime3, 10)",
    "bin_3.6" : "quantile_bin(walkD3, 10)",
    "bin_4.1" : "quantile_bin(ex4, 10)",
    "bin_4.2" : "quantile_bin(dist4, 10)",
    "bin_4.3" : "quantile_bin(minTo4, 10)",
    "bin_4.4" : "quantile_bin(travTime4, 10)",
    "bin_4.5" : "quantile_bin(exTime4, 10)",
    "bin_4.6" : "quantile_bin(walkD4, 10)",

    "car_1.1" : "cartesian(userID,cartesian(cat1, bin_1.1))",
    "car_1.2" : "cartesian(userID,cartesian(cat1, bin_1.2))",
    "car_1.3" : "cartesian(userID,cartesian(cat1, bin_1.3))",
    "car_1.4" : "cartesian(userID,cartesian(cat1, bin_1.4))",
    "car_1.5" : "cartesian(userID,cartesian(cat1, bin_1.5))",
    "car_1.6" : "cartesian(userID,cartesian(cat1, bin_1.6))",
    "car_2.1" : "cartesian(userID,cartesian(cat2, bin_2.1))",
    "car_2.2" : "cartesian(userID,cartesian(cat2, bin_2.2))",
    "car_2.3" : "cartesian(userID,cartesian(cat2, bin_2.3))",
    "car_2.4" : "cartesian(userID,cartesian(cat2, bin_2.4))",
    "car_2.5" : "cartesian(userID,cartesian(cat2, bin_2.5))",
    "car_2.6" : "cartesian(userID,cartesian(cat2, bin_2.6))",
    "car_3.1" : "cartesian(userID,cartesian(cat3, bin_3.1))",
    "car_3.2" : "cartesian(userID,cartesian(cat3, bin_3.2))",
    "car_3.3" : "cartesian(userID,cartesian(cat3, bin_3.3))",
    "car_3.4" : "cartesian(userID,cartesian(cat3, bin_3.4))",
    "car_3.5" : "cartesian(userID,cartesian(cat3, bin_3.5))",
    "car_3.6" : "cartesian(userID,cartesian(cat3, bin_3.6))",
    "car_4.1" : "cartesian(userID,cartesian(cat4, bin_4.1))",
    "car_4.2" : "cartesian(userID,cartesian(cat4, bin_4.2))",
    "car_4.3" : "cartesian(userID,cartesian(cat4, bin_4.3))",
    "car_4.4" : "cartesian(userID,cartesian(cat4, bin_4.4))",
    "car_4.5" : "cartesian(userID,cartesian(cat4, bin_4.5))",
    "car_4.6" : "cartesian(userID,cartesian(cat4, bin_4.6))",
    "CAR_GROUP1" : "group(car_1.1,car_1.2,
```

```
car_1.3,car_1.4,car_1.5,car_1.6)",
    "CAR_GROUP2" : "group(car_2.1,car_2.2,
car_2.3,car_2.4,car_2.5,car_2.6)",
    "CAR_GROUP3" : "group(car_3.1,car_3.2,
car_3.3,car_3.4,car_3.5,car_3.6)",
    "CAR_GROUP4" : "group(car_4.1,car_4.2,
car_4.3,car_4.4,car_4.5,car_4.6)"
  },
  "outputs": [
    "ALL_CATEGORICAL",
    "CAR_GROUP1",
    "CAR_GROUP2",
    "CAR_GROUP3",
    "CAR_GROUP4",
    "binned_1",
    "binned_2",
    "binned_3",
    "binned_4"
  ]
}
```

Appendix A.4 - “Cartesian” för alla möjliga delmängder för varje resa

```
{
     "groups": {
     },
     "assignments": {
          "bin_1.1" : "quantile_bin(ex1, 10)",
          "bin_1.2" : "quantile_bin(dist1, 10)",
          "bin_1.3" : "quantile_bin(minTo1, 10)",
          "bin_1.4" : "quantile_bin(travTime1, 10)",
          "bin_1.5" : "quantile_bin(exTime1, 10)",
          "bin_1.6" : "quantile_bin(walkD1, 10)",

          "car_1.1" : "cartesian(userID,cartesian(cat1, bin_1.1))",
          "car_1.2" : "cartesian(userID,cartesian(cat1, bin_1.2))",
          "car_1.3" : "cartesian(userID,cartesian(cat1, bin_1.3))",
          "car_1.4" : "cartesian(userID,cartesian(cat1, bin_1.4))",
          "car_1.5" : "cartesian(userID,cartesian(cat1, bin_1.5))",
          "car_1.6" : "cartesian(userID,cartesian(cat1, bin_1.6))",
```

```
"ac1.1-2" : "cartesian(car_1.1,v1.2)",
"ac1.1-3" : "cartesian(car_1.1,v1.3)",
"ac1.1-4" : "cartesian(car_1.1,v1.4)",
"ac1.1-5" : "cartesian(car_1.1,v1.5)",
"ac1.1-6" : "cartesian(car_1.1,v1.6)",
"ac1.2-3" : "cartesian(car_1.2,v1.3)",
"ac1.2-4" : "cartesian(car_1.2,v1.4)",
"ac1.2-5" : "cartesian(car_1.2,v1.5)",
"ac1.2-6" : "cartesian(car_1.2,v1.6)",
"ac1.3-4" : "cartesian(car_1.3,v1.4)",
"ac1.3-5" : "cartesian(car_1.3,v1.5)",
"ac1.3-6" : "cartesian(car_1.3,v1.6)",
"ac1.4-5" : "cartesian(car_1.4,v1.5)",
"ac1.4-6" : "cartesian(car_1.4,v1.6)",
"ac1.5-6" : "cartesian(car_1.5,v1.6)",

"ac1.1-2-3" : "cartesian(ac1.1-2, v1.3)",
"ac1.1-2-4" : "cartesian(ac1.1-2, v1.4)",
"ac1.1-2-5" : "cartesian(ac1.1-2, v1.5)",
"ac1.1-2-6" : "cartesian(ac1.1-2, v1.6)",
"ac1.1-3-4" : "cartesian(ac1.1-3, v1.4)",
"ac1.1-3-5" : "cartesian(ac1.1-3, v1.5)",
"ac1.1-3-6" : "cartesian(ac1.1-3, v1.6)",
"ac1.1-4-5" : "cartesian(ac1.1-4, v1.5)",
"ac1.1-4-6" : "cartesian(ac1.1-4, v1.6)",
"ac1.1-5-6" : "cartesian(ac1.1-5, v1.6)",
"ac1.2-3-4" : "cartesian(ac1.2-3, v1.4)",
"ac1.2-3-5" : "cartesian(ac1.2-3, v1.5)",
"ac1.2-3-6" : "cartesian(ac1.2-3, v1.6)",
"ac1.2-4-5" : "cartesian(ac1.2-4, v1.5)",
"ac1.2-4-6" : "cartesian(ac1.2-4, v1.6)",
"ac1.2-5-6" : "cartesian(ac1.2-5, v1.6)",
"ac1.3-4-5" : "cartesian(ac1.3-4, v1.5)",
"ac1.3-4-6" : "cartesian(ac1.3-4, v1.6)",
"ac1.3-5-6" : "cartesian(ac1.3-5, v1.6)",
"ac1.4-5-6" : "cartesian(ac1.4-5, v1.6)",

"ac1.1-2-3-4" : "cartesian(ac1.1-2-3, v1.4)",
"ac1.1-2-3-5" : "cartesian(ac1.1-2-3, v1.5)",
"ac1.1-2-3-6" : "cartesian(ac1.1-2-3, v1.6)",
"ac1.1-2-4-5" : "cartesian(ac1.1-2-4, v1.5)",
"ac1.1-2-4-6" : "cartesian(ac1.1-2-4, v1.6)",
"ac1.1-2-5-6" : "cartesian(ac1.1-2-5, v1.6)",
"ac1.1-3-4-5" : "cartesian(ac1.1-3-4, v1.5)",
```

```
          "ac1.1-3-4-6" : "cartesian(ac1.1-3-4, v1.6)",
          "ac1.1-3-5-6" : "cartesian(ac1.1-3-5, v1.6)",
          "ac1.1-4-5-6" : "cartesian(ac1.1-4-5, v1.6)",
          "ac1.2-3-4-5" : "cartesian(ac1.2-3-4, v1.5)",
          "ac1.2-3-4-6" : "cartesian(ac1.2-3-4, v1.6)",
          "ac1.2-3-5-6" : "cartesian(ac1.2-3-5, v1.6)",
          "ac1.2-4-5-6" : "cartesian(ac1.2-4-5, v1.6)",
          "ac1.3-4-5-6" : "cartesian(ac1.3-4-5, v1.6)",

          "ac1.1-2-3-4-5" : "cartesian(ac1.1-2-3-4, v1.5)",
          "ac1.1-2-3-4-6" : "cartesian(ac1.1-2-3-4, v1.6)",
          "ac1.1-2-3-5-6" : "cartesian(ac1.1-2-3-5, v1.6)",
          "ac1.1-2-4-5-6" : "cartesian(ac1.1-2-4-5, v1.6)",
          "ac1.1-3-4-5-6" : "cartesian(ac1.1-3-4-5, v1.6)",
          "ac1.2-3-4-5-6" : "cartesian(ac1.2-3-4-5, v1.6)",

          "ac1.1-2-3-4-5-6" : "cartesian(ac1.1-2-3-4-5, v1.6)"
     }
```

Appendix A.5 - “Cartesian” för att binda ihop kolumnerna mellan resorna

```
{
  "groups": {
    "GROUP_1": "group('ex1', 'dist1', 'minTo1', 'travTime1',
'exTime1', 'walkD1')",
    "GROUP_2": "group('ex2', 'dist2', 'minTo2', 'travTime2',
'exTime2', 'walkD2')"
  },
  "assignments": {

    "v1.1" : "quantile_bin(ex1,5)",
    "v1.2" : "quantile_bin(dist1,5)",
    "v1.3" : "quantile_bin(minTo1,5)",
    "v1.4" : "quantile_bin(travTime1,5)",
    "v1.5" : "quantile_bin(exTime1,5)",
    "v1.6" : "quantile_bin(walkD1,5)",
    "v2.1" : "quantile_bin(ex2,5)",
    "v2.2" : "quantile_bin(dist2,5)",
    "v2.3" : "quantile_bin(minTo2,5)",
    "v2.4" : "quantile_bin(travTime2,5)",
    "v2.5" : "quantile_bin(exTime2,5)",
```

```
    "v2.6" : "quantile_bin(walkD2,5)",

    "ex1-2" : "cartesian(v1.1, v2.1)",
    "dist1-2" : "cartesian(v1.2,v2.2)",
    "minTo1-2" : "cartesian(v1.3,v2.3)",
    "travTime1-2" : "cartesian(v1.4,v2.4)",
    "exTime1-2" : "cartesian(v1.5,v2.5)",
    "walkD1-2" : "cartesian(v1.6, v2.6)",

    "crossgroup" : "group(ex1-2,dist1-2, minTo1-2, travTime1-2,
exTime1-2, walkD1-2)"

  },
  "outputs": [
    "GROUP_1",
    "GROUP_2",
    "crossgroup"
  ]
}
```

Appendix A.6 - “Cartesian” för att binda ihop kolumnerna mellan resorna och internt

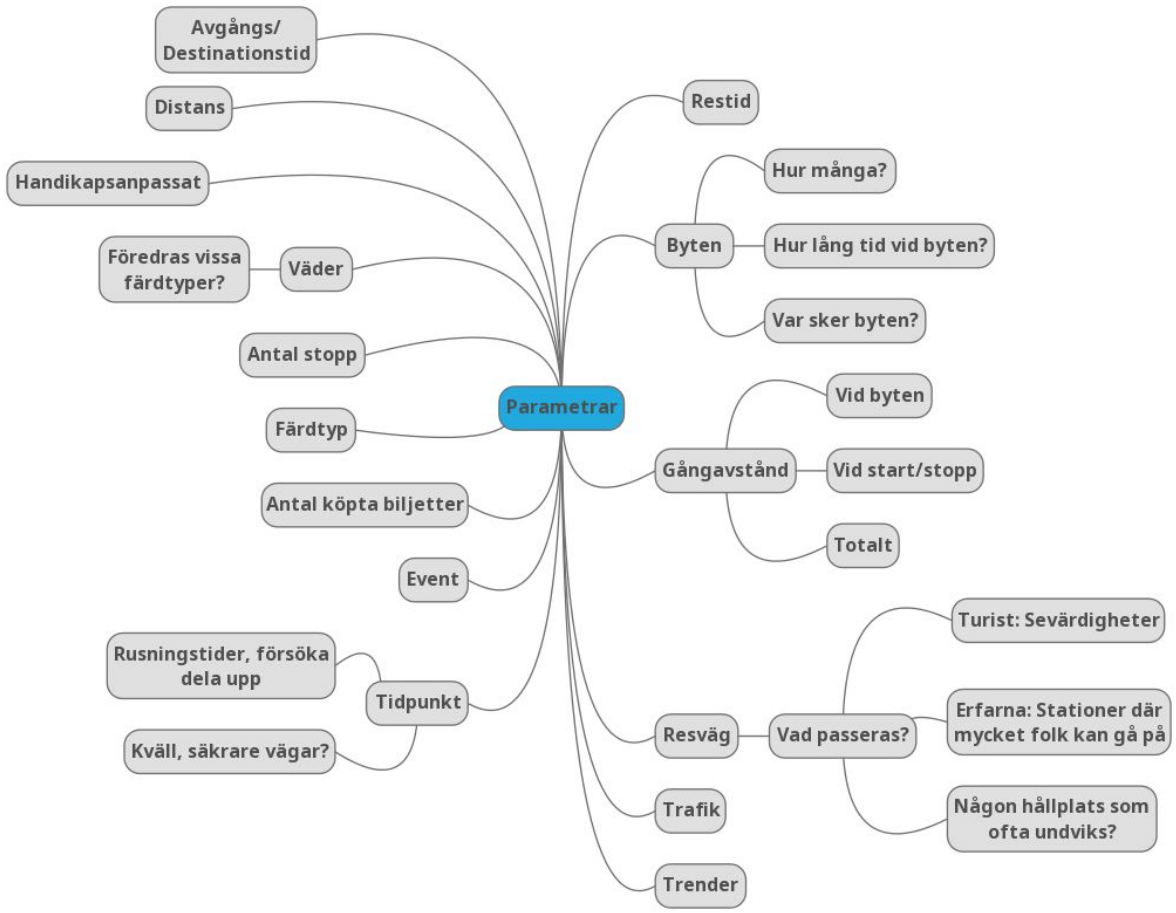
```
{
  "groups": {
    "GROUP_1": "group('ex1', 'dist1', 'minTo1', 'travTime1',
'exTime1', 'walkD1')",
    "GROUP_2": "group('ex2', 'dist2', 'minTo2', 'travTime2',
'exTime2', 'walkD2')"
  },
  "assignments": {

    "v1.1" : "quantile_bin(ex1,5)",
    "v1.2" : "quantile_bin(dist1,5)",
    "v1.3" : "quantile_bin(minTo1,5)",
    "v1.4" : "quantile_bin(travTime1,5)",
    "v1.5" : "quantile_bin(exTime1,5)",
    "v1.6" : "quantile_bin(walkD1,5)",
    "v2.1" : "quantile_bin(ex2,5)",
    "v2.2" : "quantile_bin(dist2,5)",
```

```
    "v2.3" : "quantile_bin(minTo2,5)",
    "v2.4" : "quantile_bin(travTime2,5)",
    "v2.5" : "quantile_bin(exTime2,5)",
    "v2.6" : "quantile_bin(walkD2,5)",

    "c1.1-2" : "cartesian(v1.1, v1.2)",
    "c1.1-3" : "cartesian(v1.1, v1.3)",
    "c1.1-4" : "cartesian(v1.1, v1.4)",
    "c1.1-5" : "cartesian(v1.1, v1.5)",
    "c1.1-6" : "cartesian(v1.1, v1.6)",
    "c1.2-3" : "cartesian(v1.2, v1.3)",
    "c1.2-4" : "cartesian(v1.2, v1.4)",
    "c1.2-5" : "cartesian(v1.2, v1.5)",
    "c1.2-6" : "cartesian(v1.2, v1.6)",
    "c1.3-4" : "cartesian(v1.3, v1.4)",
    "c1.3-5" : "cartesian(v1.3, v1.5)",
    "c1.3-6" : "cartesian(v1.3, v1.6)",
    "c1.4-5" : "cartesian(v1.4, v1.5)",
    "c1.4-6" : "cartesian(v1.4, v1.6)",
    "c1.5-6" : "cartesian(v1.5, v1.6)",
    "c2.1-2" : "cartesian(v2.1, v2.2)",
    "c2.1-3" : "cartesian(v2.1, v2.3)",
    "c2.1-4" : "cartesian(v2.1, v2.4)",
    "c2.1-5" : "cartesian(v2.1, v2.5)",
    "c2.1-6" : "cartesian(v2.1, v2.6)",
    "c2.2-3" : "cartesian(v2.2, v2.3)",
    "c2.2-4" : "cartesian(v2.2, v2.4)",
    "c2.2-5" : "cartesian(v2.2, v2.5)",
    "c2.2-6" : "cartesian(v2.2, v2.6)",
    "c2.3-4" : "cartesian(v2.3, v2.4)",
    "c2.3-5" : "cartesian(v2.3, v2.5)",
    "c2.3-6" : "cartesian(v2.3, v2.6)",
    "c2.4-5" : "cartesian(v2.4, v2.5)",
    "c2.4-6" : "cartesian(v2.4, v2.6)",
    "c2.5-6" : "cartesian(v2.5, v2.6)",

    "c1.1-2-3" : "cartesian(c1.1-2, v1.3)",
    "c2.1-2-3" : "cartesian(c2.1-2, v2.3)",

    "c1.1-2-3-4" : "cartesian(c1.1-2-3, v1.4)",
    "c2.1-2-3-4" : "cartesian(c2.1-2-3, v2.4)",

    "c1.1-2-3-4-5" : "cartesian(c1.1-2-3-4, v1.5)",
    "c2.1-2-3-4-5" : "cartesian(c2.1-2-3-4, v2.5)",
```

```
    "c1.1-2-3-4-5-6" : "cartesian(c1.1-2-3-4-5, v1.6)",
    "c2.1-2-3-4-5-6" : "cartesian(c2.1-2-3-4-5, v2.6)",

    "c1" : "cartesian(v1.1, v2.1)",
    "c2" : "cartesian(v1.2, v2.2)",
    "c3" : "cartesian(v1.3, v2.3)",
    "c4" : "cartesian(v1.4, v2.4)",
    "c5" : "cartesian(v1.5, v2.5)",
    "c6" : "cartesian(v1.6, v2.6)",

    "c1-2" : "cartesian(c1.1-2, c2.1-2)",
    "c1-3" : "cartesian(c1.1-3, c2.1-3)",
    "c1-4" : "cartesian(c1.1-4, c2.1-4)",
    "c1-5" : "cartesian(c1.1-5, c2.1-5)",
    "c1-6" : "cartesian(c1.1-6, c2.1-6)",
    "c2-3" : "cartesian(c1.2-3, c2.2-3)",
    "c2-4" : "cartesian(c1.2-4, c2.2-4)",
    "c2-5" : "cartesian(c1.2-5, c2.2-5)",
    "c2-6" : "cartesian(c1.2-6, c2.2-6)",
    "c3-4" : "cartesian(c1.3-4, c2.3-4)",
    "c3-5" : "cartesian(c1.3-5, c2.3-5)",
    "c3-6" : "cartesian(c1.3-6, c2.3-6)",
    "c4-5" : "cartesian(c1.4-5, c2.4-5)",
    "c4-6" : "cartesian(c1.4-6, c2.4-6)",
    "c5-6" : "cartesian(c1.5-6, c2.5-6)",

    "crossgroup" : "group(c1,c2,c3,c4,c5,c6,c1-2, c1-3, c1-4, c1-5,
c1-6, c2-3, c2-4, c2-5, c2-6, c3-4, c3-5, c3-6, c4-5, c4-6, c5-6)"
  },
  "outputs": [
    "GROUP_1",
    "GROUP_2",
    "crossgroup",
    "c1.1-2-3-4-5-6",
    "c2.1-2-3-4-5-6"
  ]
}
```

Appendix B - Maskininlärningsmodeller

Appendix B.1 - Parametrar efter första analysen



Appendix B.2 - Recept för maskininlärningsmodellerna

```
{
  "groups": {
    "GROUP_1": "group('ex1', 'dist1', 'minTo1', 'travTime1', 'exTime1',
'walkD1')",
    "GROUP_2": "group('ex2', 'dist2', 'minTo2', 'travTime2', 'exTime2',
'walkD2')"
  },
  "assignments": {
    "v1.1" : "quantile_bin(ex1,5)",
    "v1.2" : "quantile_bin(dist1,5)",
```

```
    "v1.3" : "quantile_bin(minTo1,5)",
    "v1.4" : "quantile_bin(travTime1,5)",
    "v1.5" : "quantile_bin(exTime1,5)",
    "v1.6" : "quantile_bin(walkD1,5)",

    "c1.1-2" : "cartesian(v1.1, v1.2)",
    "c1.1-3" : "cartesian(v1.1, v1.3)",
    "c1.1-4" : "cartesian(v1.1, v1.4)",
    "c1.1-5" : "cartesian(v1.1, v1.5)",
    "c1.1-6" : "cartesian(v1.1, v1.6)",
    "c1.2-3" : "cartesian(v1.2, v1.3)",
    "c1.2-4" : "cartesian(v1.2, v1.4)",
    "c1.2-5" : "cartesian(v1.2, v1.5)",
    "c1.2-6" : "cartesian(v1.2, v1.6)",
    "c1.3-4" : "cartesian(v1.3, v1.4)",
    "c1.3-5" : "cartesian(v1.3, v1.5)",
    "c1.3-6" : "cartesian(v1.3, v1.6)",
    "c1.4-5" : "cartesian(v1.4, v1.5)",
    "c1.4-6" : "cartesian(v1.4, v1.6)",
    "c1.5-6" : "cartesian(v1.5, v1.6)",

    "c1.1-2-3" : "cartesian(c1.1-2, v1.3)",
    "c1.1-2-3-4" : "cartesian(c1.1-2-3, v1.4)",
    "c1.1-2-3-4-5" : "cartesian(c1.1-2-3-4, v1.5)",
    "c1.1-2-3-4-5-6" : "cartesian(c1.1-2-3-4-5, v1.6)",

    // Motsvarande för variablerna i “GROUP_2”

    "c1" : "cartesian(v1.1, v2.1)",
    "c2" : "cartesian(v1.2, v2.2)",
    "c3" : "cartesian(v1.3, v2.3)",
    "c4" : "cartesian(v1.4, v2.4)",
    "c5" : "cartesian(v1.5, v2.5)",
    "c6" : "cartesian(v1.6, v2.6)",
    "c1-2" : "cartesian(c1.1-2, c2.1-2)",
    "c1-3" : "cartesian(c1.1-3, c2.1-3)",
    "c1-4" : "cartesian(c1.1-4, c2.1-4)",
    "c1-5" : "cartesian(c1.1-5, c2.1-5)",
    "c1-6" : "cartesian(c1.1-6, c2.1-6)",
    "c2-3" : "cartesian(c1.2-3, c2.2-3)",
    "c2-4" : "cartesian(c1.2-4, c2.2-4)",
    "c2-5" : "cartesian(c1.2-5, c2.2-5)",
    "c2-6" : "cartesian(c1.2-6, c2.2-6)",
    "c3-4" : "cartesian(c1.3-4, c2.3-4)",
    "c3-5" : "cartesian(c1.3-5, c2.3-5)",
    "c3-6" : "cartesian(c1.3-6, c2.3-6)",
    "c4-5" : "cartesian(c1.4-5, c2.4-5)",
```

```
    "c4-6" : "cartesian(c1.4-6, c2.4-6)",
    "c5-6" : "cartesian(c1.5-6, c2.5-6)",
    "crossgroup" : "group(c1,c2,c3,c4,c5,c6,c1-2, c1-3, c1-4, c1-5, c1-6,
              c2-3, c2-4, c2-5, c2-6, c3-4, c3-5, c3-6, c4-5, c4-6, c5-6)"
  },
  "outputs": [
    "GROUP_1",
    "GROUP_2",
    "crossgroup",
    "c1.1-2-3-4-5-6",
    "c2.1-2-3-4-5-6"
  ]
}
```

Appendix B.3 - Testning av maskininlärningsmodellerna

I kapitlet presenteras resultaten för testerna av maskininlärningsmodellerna.

Appendix B.3.1 - Evaluering av maskininlärningsmodellerna

I kapitlet visas resultaten från evalueringen av maskininlärningsmodellerna i Amazon Machine Learning.

Tabell B.1 - Tabell som visar precisionen för maskininlärningsmodellerna för de olika användarna och den genomsnittliga precisionen

Antal rader i träningsdatan	Precision för resenär 1	Precision för resenär 2	Precision för resenär 3	Precision för resenär 4	Genomsnittlig precision
1000	96,4%	98,9%	97,9%	97,4%	98,0%
400	94,8%	97,1%	96,8%	96,1%	96,2%
100	85,3%	93,6%	91,8%	93,2%	91,0%

I Tabell B.1 finner man att den genomsnittliga precisionen ligger över 90% för maskininlärningsmodeller med träningsdata i alla testade storlekar. Det resultat som sticker ut är precisionen på maskininlärningsmodellen med träningsdatan för resenär 1 med 100 rader. Anledningen till detta kan bero på att träningsdatan eller evalueringsdatan är svår att tyda för maskininlärningsmodellen.

Tabell B.2 - Tabell som visar precisionen för en maskininlärningsmodell där resenär 3 successivt byter över till resenär 4. I tabellen presenteras även differensen mellan precisionen för resenär 3 och resenär 4

Antal rader från resenär 3	Antal rader från resenär 4	Precision på evalueringsdatan för resenär 3	Precision för evalueringsdatan för resenär 4	Differens mellan precisionen för resenär 3 och resenär 4
90	10	91,7%	60,2%	31,5
80	20	87,7%	64,6%	23,1%
70	30	85,3%	66,6%	18,7%
60	40	78,4%	70,3%	8,1%
50	50	78,6%	77,0%	1,6%
40	60	75,5%	81,3%	-5,8%
30	70	72,7%	84,6%	-11,9%
20	80	69,4%	88,6%	-19,2%
10	90	67,4%	89,3%	-21,9
0	100	55,5%	93,2%	-37,7

I Tabell B.2 kan man se hur precisionen påverkas när en resenär byter preferenser. I tabellen visas övergången från preferenserna för resenär 3 till preferenserna för resenär 4. Precisionen presenteras genom en evaluering på evalueringsdatan för resenär 3 och en evaluering på evalueringsdatan för resenär 4. Man kan även utläsa differensen mellan precisionerna.

Som förväntat sjunker precisionen för resenär 3s evalueringsdata och stiger för resenär 4s evalueringsdata. Övergången, då precisionen för resenär 4 är högre än precisionen för resenär 3 sker när datasetet innehåller 60 rader från resenär 4s preferenser vilket är förväntat. Något som är intressant är att precisionen överstiger 80% redan vid 60 rader med preferenser från resenär 4, vilket är en relativt bra precision.

Appendix B.3.2 - Ingen maskininlärning

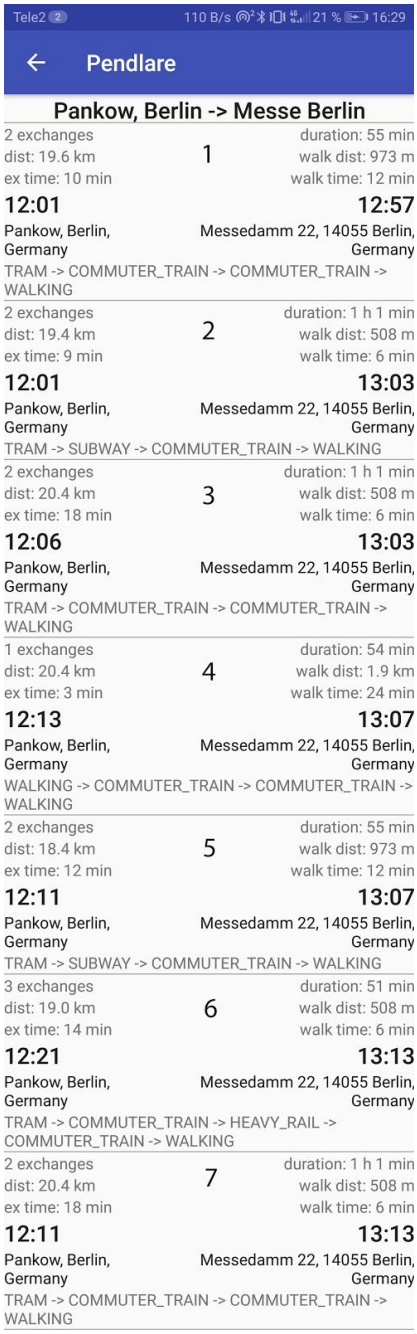


Fig. B.1 - Listan på resor när ingen maskininlärning har applicerats.

Figur B.1 visar listan på resorna när ingen maskininlärning appliceras. Resorna är numrerade från 1 till 7 för att enklare kunna referera till dessa i kommande tester.

Appendix B.3.3 - Endast global maskininlärningsmodell (ny resenär)

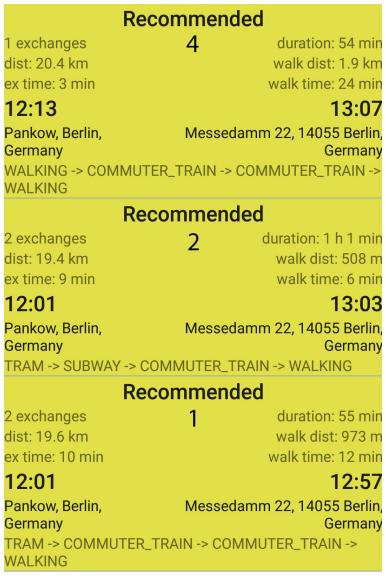


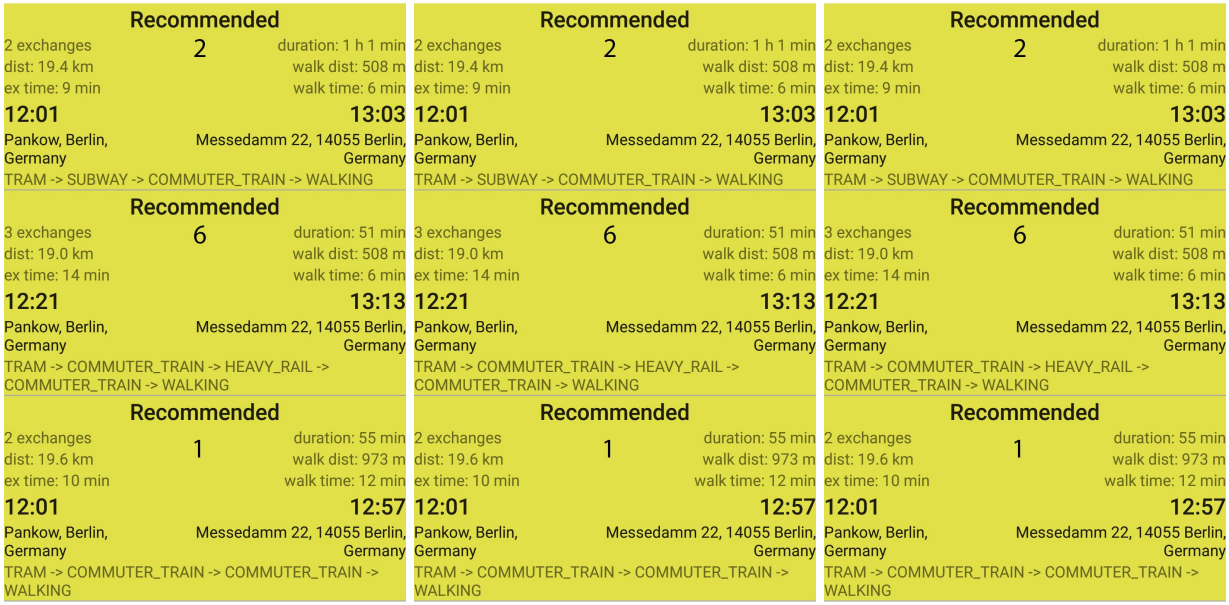
Fig. B.2 - Endast den globala maskininlärningsmodellen

I Figur B.2 visas resultatet då endast den globala maskininlärningsmodellen används. Då två av resenärerna, resenär 2 och 3, har preferenser rörande byten (tid vid byten och antal byten) är det förväntat att dessa preferenser har störst påverkan. Antal minuter till avgång har också en stor påverkan men inte lika stor som preferenserna rörande byten. Övriga preferenser är svåra att urskilja exakt, men vid jämförelse av resa 2 och 1 har de samma antal minuter till avgång. Resa 2 har däremot lägre värde på distans, tid vid byten och gångavstånd, det vill säga alla andra preferenser. Därför är det förväntat att den har större sannolikhet att väljas av resenären jämfört med resa 1. Det är däremot svårt att urskilja exakt vilken/vilka parametrar som har störst påverkan för resultatet.

Appendix B.3.4 - Tester för resenärer med preferenser som inte ändras

I följande tester visas resultaten för de olika resenärerna. Resultaten är uppdelade i tre delar, en del där alla maskininlärningsmodeller är aktiva, en när endast den stora och den lilla maskininlärningsmodellen är aktiv och en där endast den lilla maskininlärningsmodellen är aktiv.

Resenär 1



Globala, stora och lilla | Stora och lilla | Endast lilla

Fig. B.3 - Resultaten för resenär 1

Resultaten som visas i Figur B.3 för resenär 1 stämmer ganska bra överens med det förväntade resultatet. Resa 2 har lägst antal minuter till avgång och tredje lägst distans, vilket speglar resenärens preferenser. Resa 6 har näst lägst distans men mest antal minuter till avgång, och skulle enligt preferenserna kunnat bytas ut mot resa 5 istället. Resa 1 har lägst antal minuter till avgång och låg distans.

Resenär 2

Globala, stora och lilla | Stora och lilla | Endast lilla

Fig. B.4 - Resultaten för resenär 2

Resultaten som visas i Figur B.4 för resenär 2 stämmer bra överens med resenärens preferenser. Resa 4 har avsevärt lägst tid vid byten och inte allt för högt antal minuter till avgång. Resa 2 och 1 har lägst antal minuter till avgång och näst respektive tredje lägst tid vid byten.

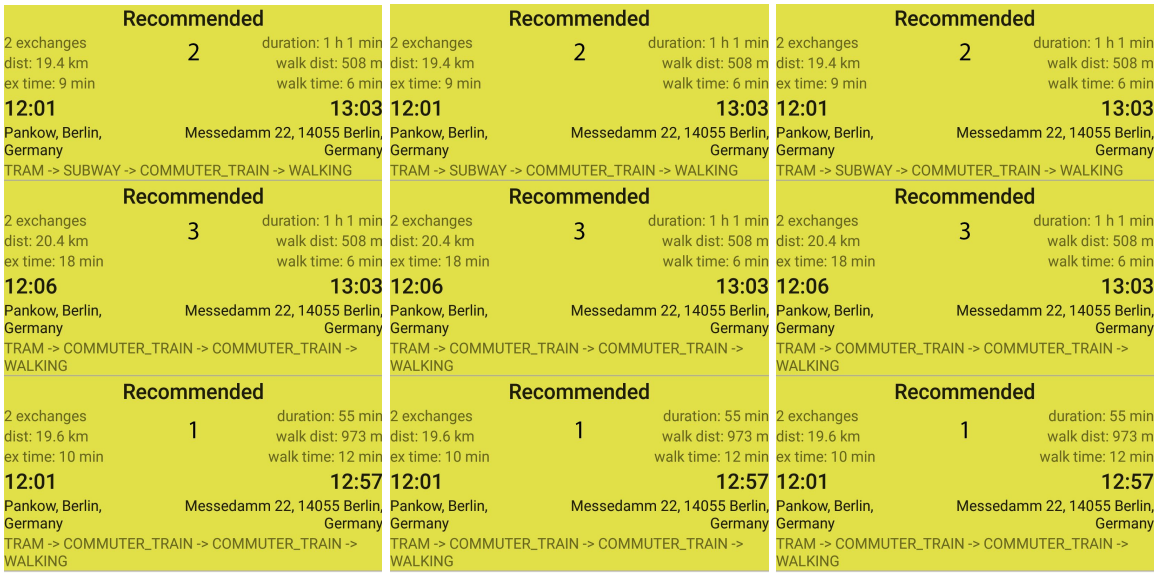
Resenär 3

Globala, stora och lilla | Stora och lilla | Endast lilla

Fig. B.5 - Resultaten för resenär 3

Resultaten som visas i Figur B.5 för resenär 3 stämmer sådär bra överens med preferenserna. Resa 4 och 2 är på de förväntade platserna medans resa 7 bör bytas ut mot resa 1 då dessa har samma antal byten men resa 1 har lägre antal minuter till avgång.

Resenär 4



Globala, stora och lilla | Stora och lilla | Endast lilla

Fig. B.6 - Resultat för resenär 4

Resultaten i Figur B.6 för resenär 4 stämmer överens med resenärens preferenser. Resa 2 och 3 har lägst gångavstånd och resa 2 har lägst antal minuter till avgång medan 3 har näst lägst. Resa 1 har lägst antal minuter till avgång samt näst lägst gångavstånd.

Slutsats

I det här testet visades förmågan hos maskininlärningsmodellerna att förutsäga vilka resor en resenär mest sannolikt kommer välja. På vissa resor gör den i viss mån felaktiga förutsägelser men överlag presterar den bra.

Appendix B.3.5 - Tester för när resenärer har bytt preferenser

I följande tester visas resultaten för när en resenär har en sorts preferens i det stora datasetet och en annan i den lilla datasetet. Syftet med testet är att se hur maskininlärningsmodellen presterar då en resenärs preferenser har ändrats.

Lilla datasetet från resenär 1, stora datasetet från resenär 2

Fig. B.7 - Resultat för en resenär med det lilla datasetet från resenär 1 och det stora datasetet från resenär 2

Resultatet i Figur B.7 stämmer överens med vad som kan förväntas. Resultaten innehåller de resor som resenär 1 fick i sina resultat men resa 1 och 6 har bytt plats då resa 1 har lägre tid vid byten.

Lilla datasetet från resenär 2, stora datasetet från resenär 3

Globala, stora och lilla Stora och lilla

Fig. B.8 - Resultat för en resenär med det lilla datasetet från resenär 2 och det stora datasetet från resenär 3

Resultatet i Figur B.8 blir likadant som för resenär 2. Det är förväntat då båda resenärerna har byten som högst prioriterade preferens. Resenär 2 har tid vid byten och resenär 3 har antal byten, vilket i det här fallet har ett samband.

Lilla datasetet från resenär 3, stora datasetet från resenär 4

Globala, stora och lilla Stora och lilla

Fig. B.9 - Resultat för en resenär med det lilla datasetet från resenär 3 och det stora datasetet från resenär 4

I Figur B.9 kan man se att de presenterade resorna är samma resor som presenterades i resultatet för resenär 3, men resa 2 och 7 har bytt plats. Detta är förväntat då resenär 4 har antal minuter till avgång som högsta prioritet och resa 2 har lägst antal minuter till avgång. Som i testet för resenär 3 bör resa 7 bytas ut till resa 1 för optimalt resultat.

Lilla datasetet från resenär 4, stora datasetet från resenär 1

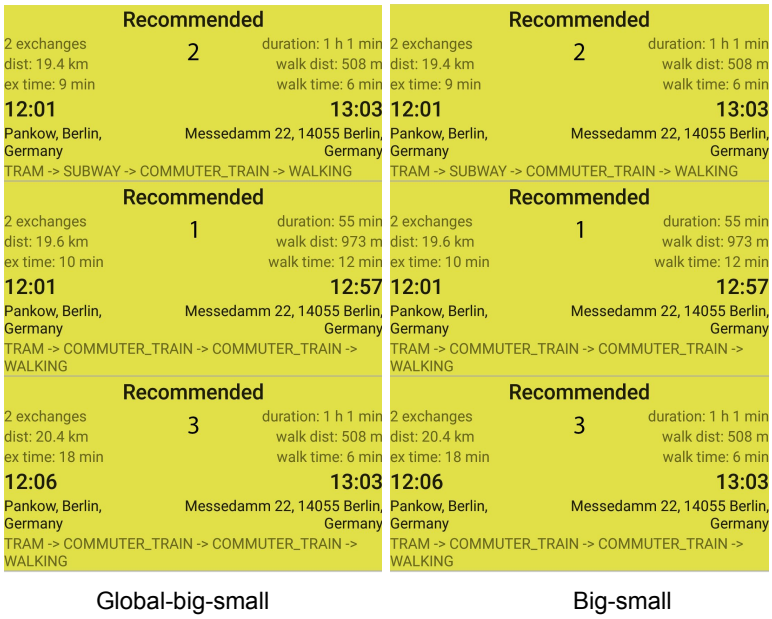


Fig. B.10 - Resultat för en resenär med det lilla datasetet från resenär 4 och det stora datasetet från resenär 1

I Figur B.10 kan man även här se att alla resor som presenterades i resultatet för resenär 4 finns med, men att resa 1 och 3 har bytt plats. Då resa 1 både har lägre distans och antal minuter till avgång, vilka båda finns i preferenserna för resenär 1 är det förväntat att den ska flyttas upp, även om den har längre gångavstånd än resa 3.

Slutsats

Genom dessa testerna kan slutsatsen dras om att maskininlärningsmodellen kan förutsäga de förväntade resorna när en resenär har bytt ut sina preferenser. De gamla preferenserna påverkar fortfarande resultatet i viss mån men det är huvudsakligen de nya preferenserna som används.

Appendix B.3.6 - Tester under tiden en resenär byter preferenser

För att testa hur responsiv maskininlärningsmodellen är för förändringar i preferenser testades resultaten för när en resenär börjar välja resor baserat på nya preferenser. I det här testet testades det hur resultaten presenteras när resenär 3 successivt börjar välja resor med preferenser som resenär 4. I varje steg byts 20 rader (cirka 3 resor) ut i det lilla datasetet och samma resor läggs till i det stora datasetet.

100 rader från resenär 3, 0 rader från resenär 4

Global-big-small Big-small Small

Fig. B.11 - Resultat för när resenär 3 har valt 0 resor med preferenser som resenär 4

I Figur B.11 visas utgångspunkten. Resultaten som presenteras här är samma som för resenär 3.

80 rader från resenär 3, 20 rader från resenär 4

Fig. B.12 - Resultat för när resenär 3 har valt 20 rader med preferenser som resenär 4

I Figur B.12 kan man se att resultaten har påverkats av dessa 20 nya raderna. I resultaten där den globala, stora och lilla maskininlärningsmodellen har använts har resultatet ändrats så att resa 7 har bytts ut mot resa 1, som har lika många byten men ett lägre antal minuter till avgång. I resultatet där endast den lilla maskininlärningsmodellen använts presenteras samma resultat som presenteras i resultatet för när 0 rader har valts med preferenser som resenär 4, men med en annan ordning. Resa 2 flyttades upp till toppen då den har lägst antal minuter till avgång medans resa 7 flyttades ner till botten då den har ett högre antal minuter till avgång än resa 2 och ett högre antal byten än resa 4.

60 rader från resenär 3, 40 rader från resenär 4

Fig. B.13 - Resultat för när resenär 3 har valt 40 rader med preferenser som resenär 4

I Figur B.13 ser man att antalet minuter till avgång fokuseras mer. Antalet byten har fortfarande en påverkan vilket man kan se på resa 4 då den har ett högre antal minuter till avgångstid än vad till exempel resa 3 har men lägre antal byten. På resa 2 och 1 ser man även att gångavståndet påverkar ordningen då resa 2 har lägre gångavstånd än resa 1, men samma antal minuter till avgång.

40 rader från resenär 3, 60 rader från resenär 4

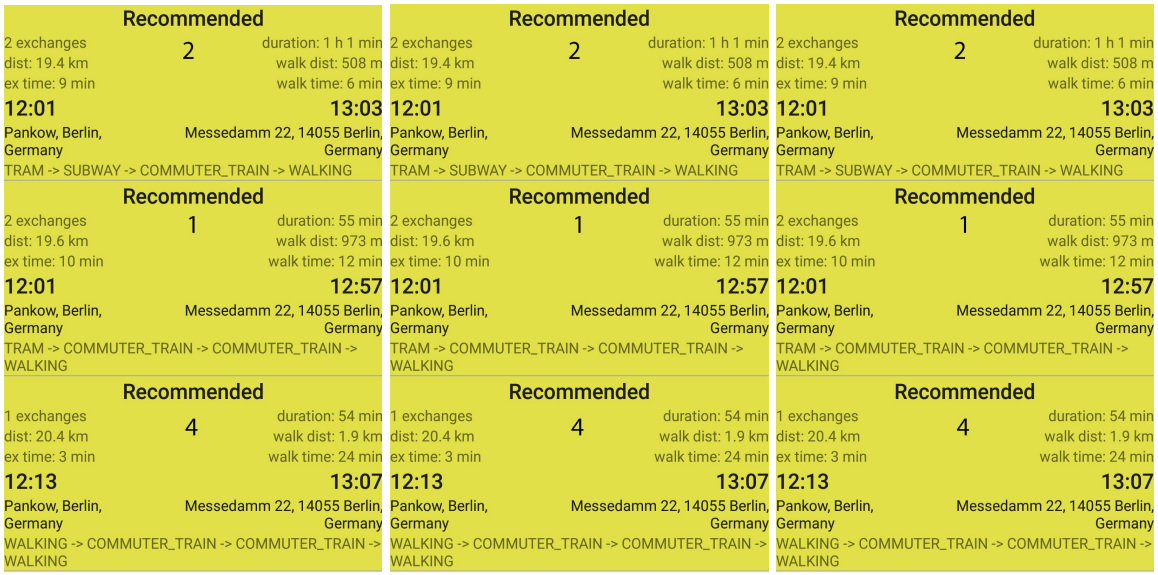


Fig. B.14 - Resultat för när resenär 3 har valt 60 rader med preferenser som resenär 4

I Figur B.14 har ingen förändring skett från föregående steg.

20 rader från resenär 3, 80 rader från resenär 4

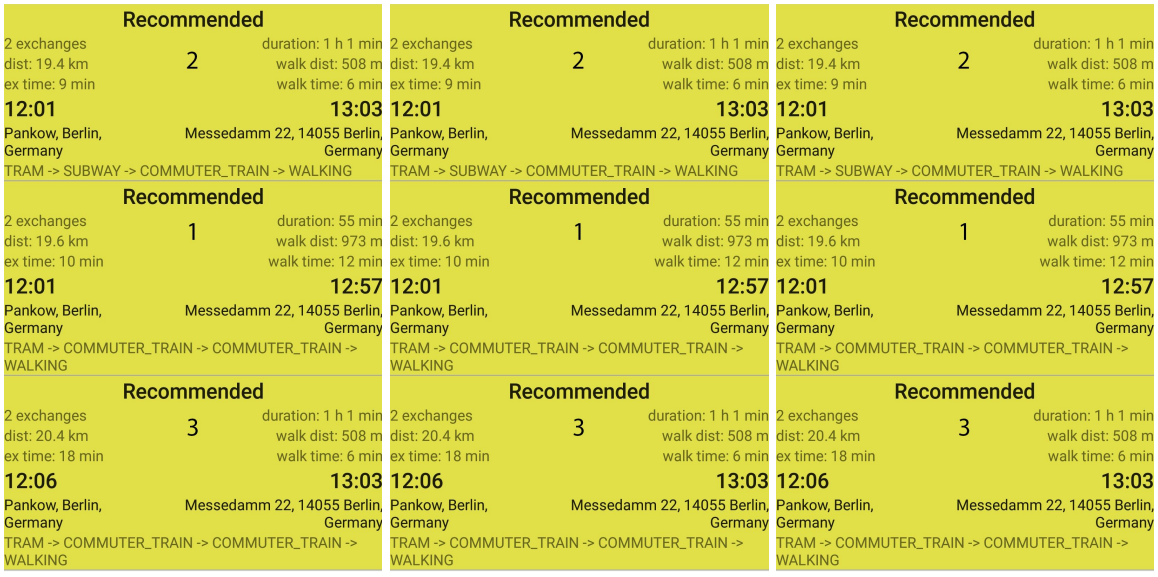


Fig. B.15 - Resultat för när resenär 3 har valt 80 rader med preferenser som resenär 4

I Figur B.15 kan man se att antalet byten inte längre fokuseras utan preferenserna från resenär 4 är de som fokuseras. Man kan även se att preferensen från resenär 3 för antalet minuter till avgång fortfarande påverkar då resorna är sorterade utifrån denna preferens i första hand och vid lika antal minuter till avgångstid är det gångavståndet som avgör.

0 rader från resenär 3, 100 rader från resenär 4

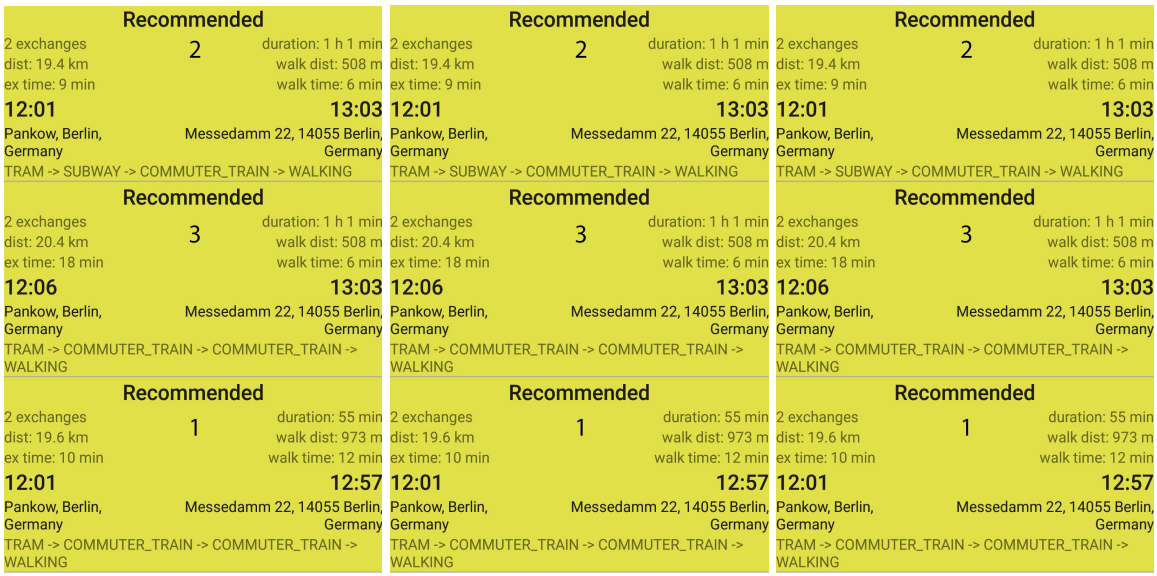


Fig. B.16 - Resultatet för när resenär 3 har valt 100 rader med preferenser som resenär 4

I Figur B.16 kan man se att det nu endast är preferenserna från resenär 4 som används genom att jämföra med resultatet från testerna på resenär 4.

Slutsats

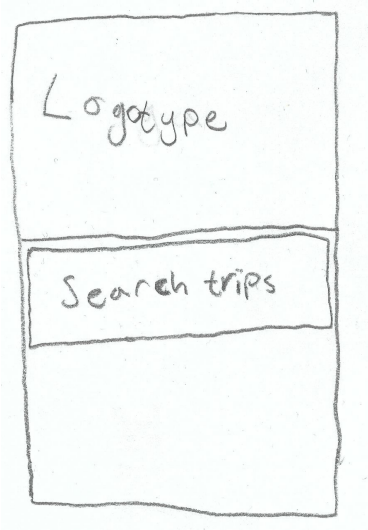
Utifrån det här testet kan slutsatsen dras att maskininlärningsmodellerna snabbt reagerar på förändring av preferenser. Redan efter 20 rader (cirka 3 resor) med en annan preferens börjar resultaten visa inslag av den nya preferensen. Efter 40 rader (cirka 6 resor) har påverkan av de nya preferenserna blivit tillräckligt stor för att börja ta över de gamla preferenserna. Efter att 80 rader (cirka 11-12 resor) har valts har de gamla preferenserna endast en liten påverkan på resultatet och efter 100 rader (cirka 15 resor) har de tagit över helt.

Appendix C - Gränssnitt för prototyp

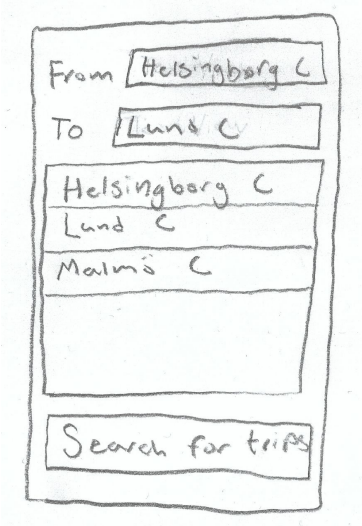
Appendix C.1 - Pappersprototyp

Nedan visas bilder på pappersprototypen som gränssnitten i Android prototypen baseras på.

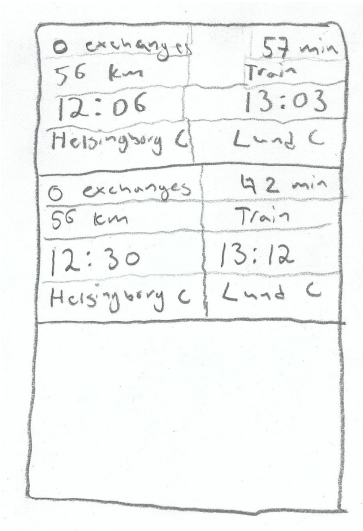
Startsida



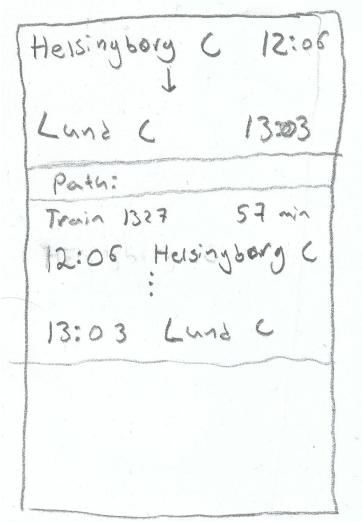
Söksida



Resultatsida



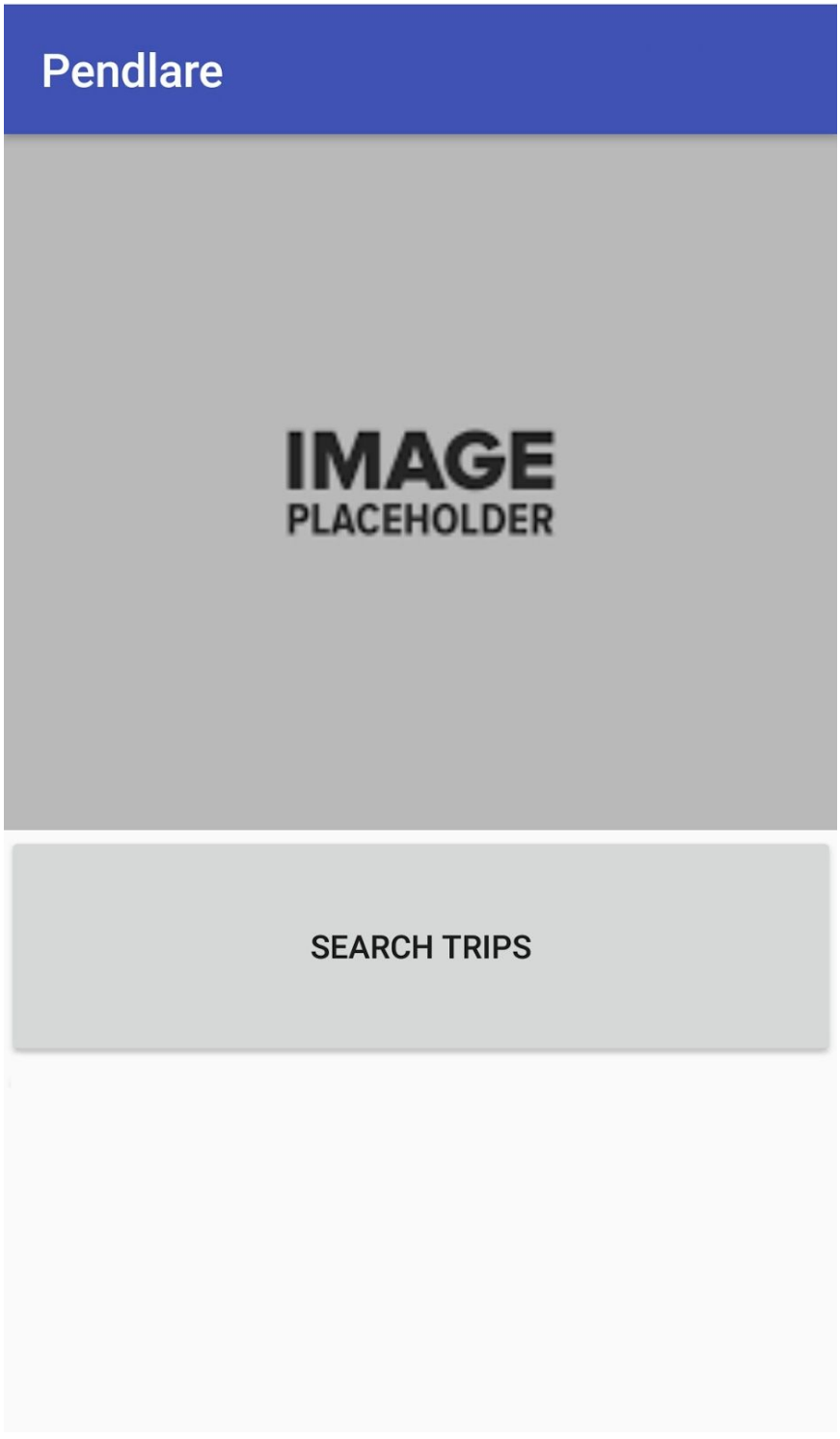
Specifik resultatsida



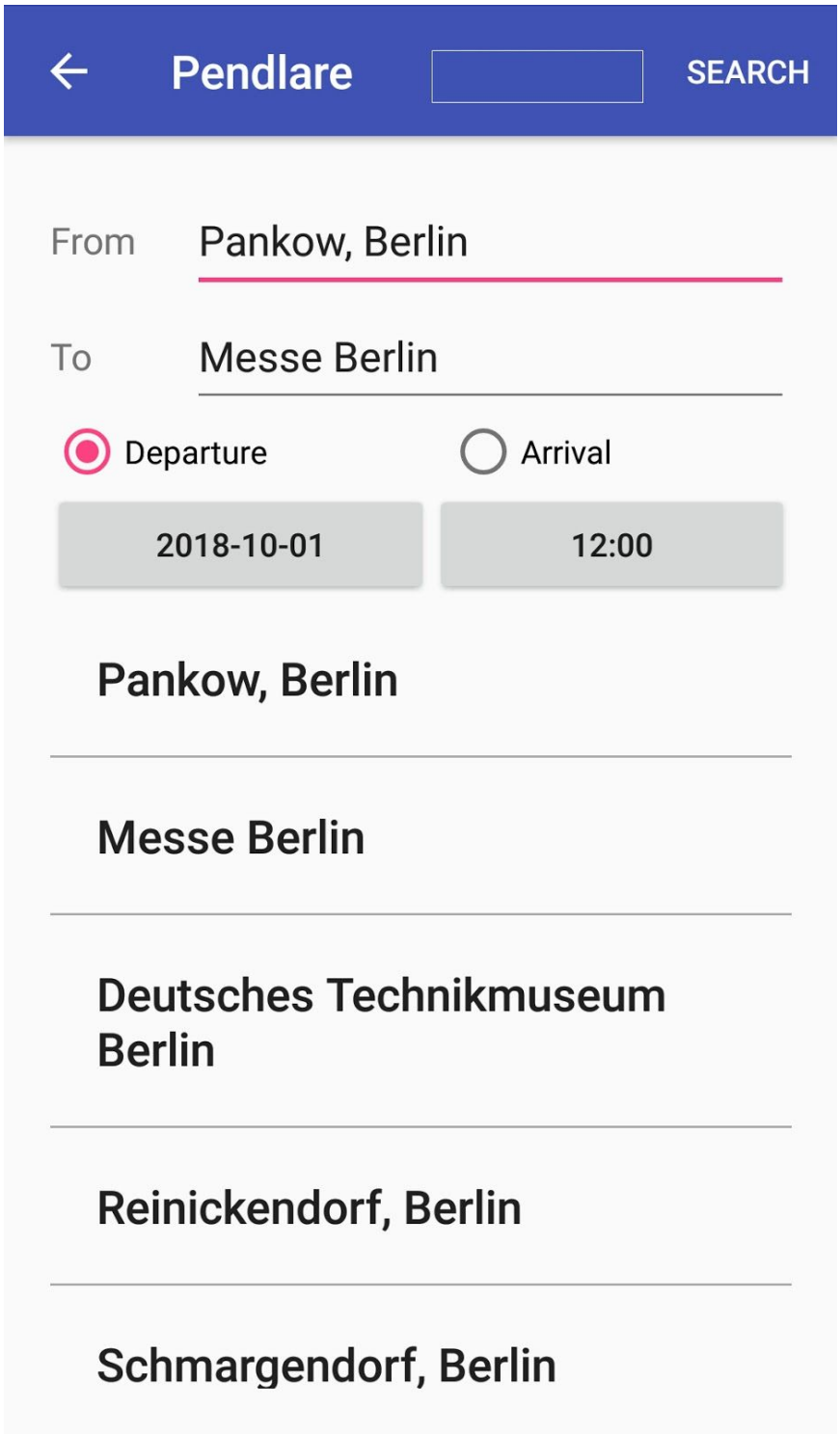
Appendix C.2 - Gränssnitt i Android-prototypen

Nedan visas bilder på gränssnitten i Android-prototypen.

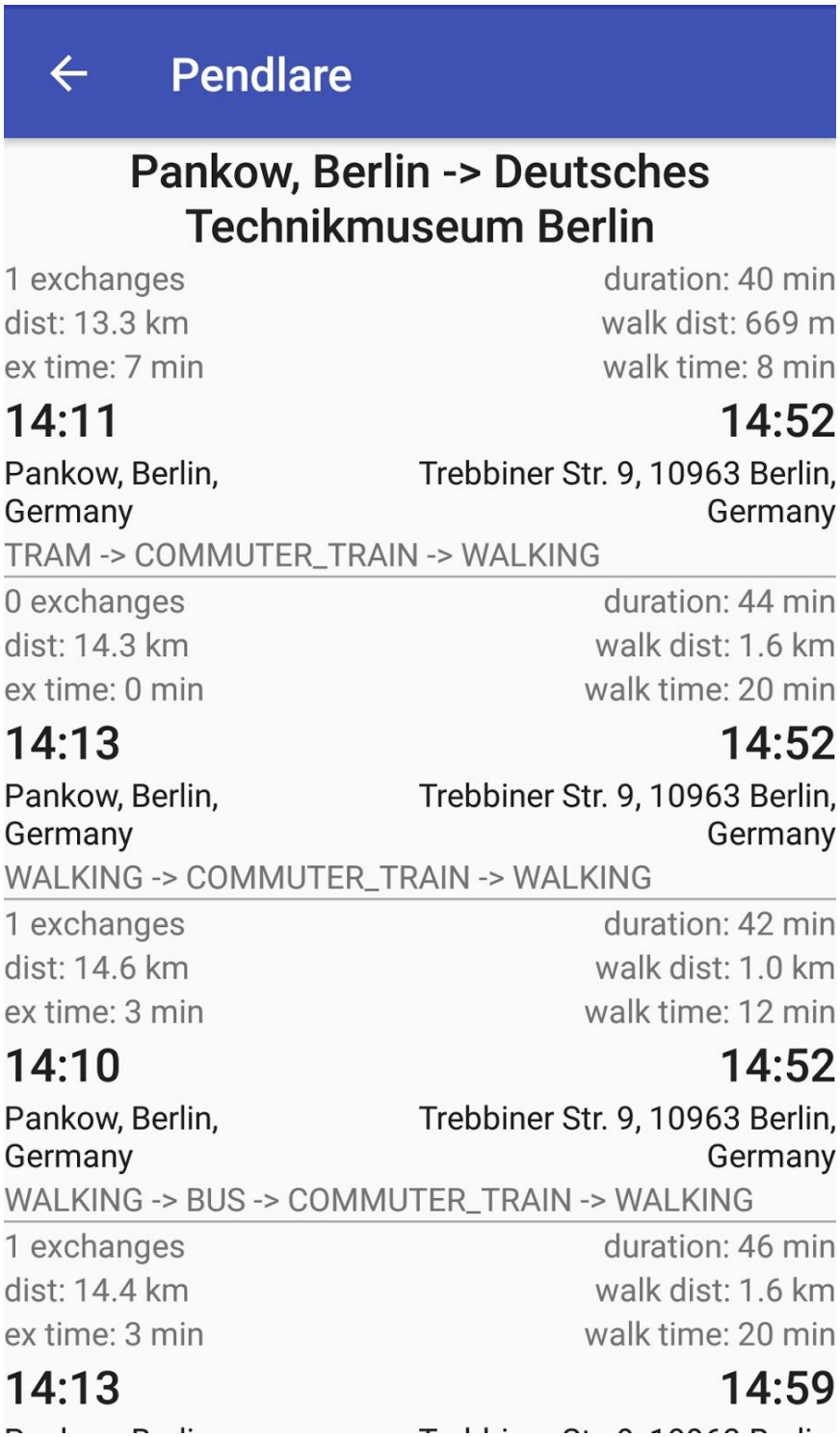
Startsida



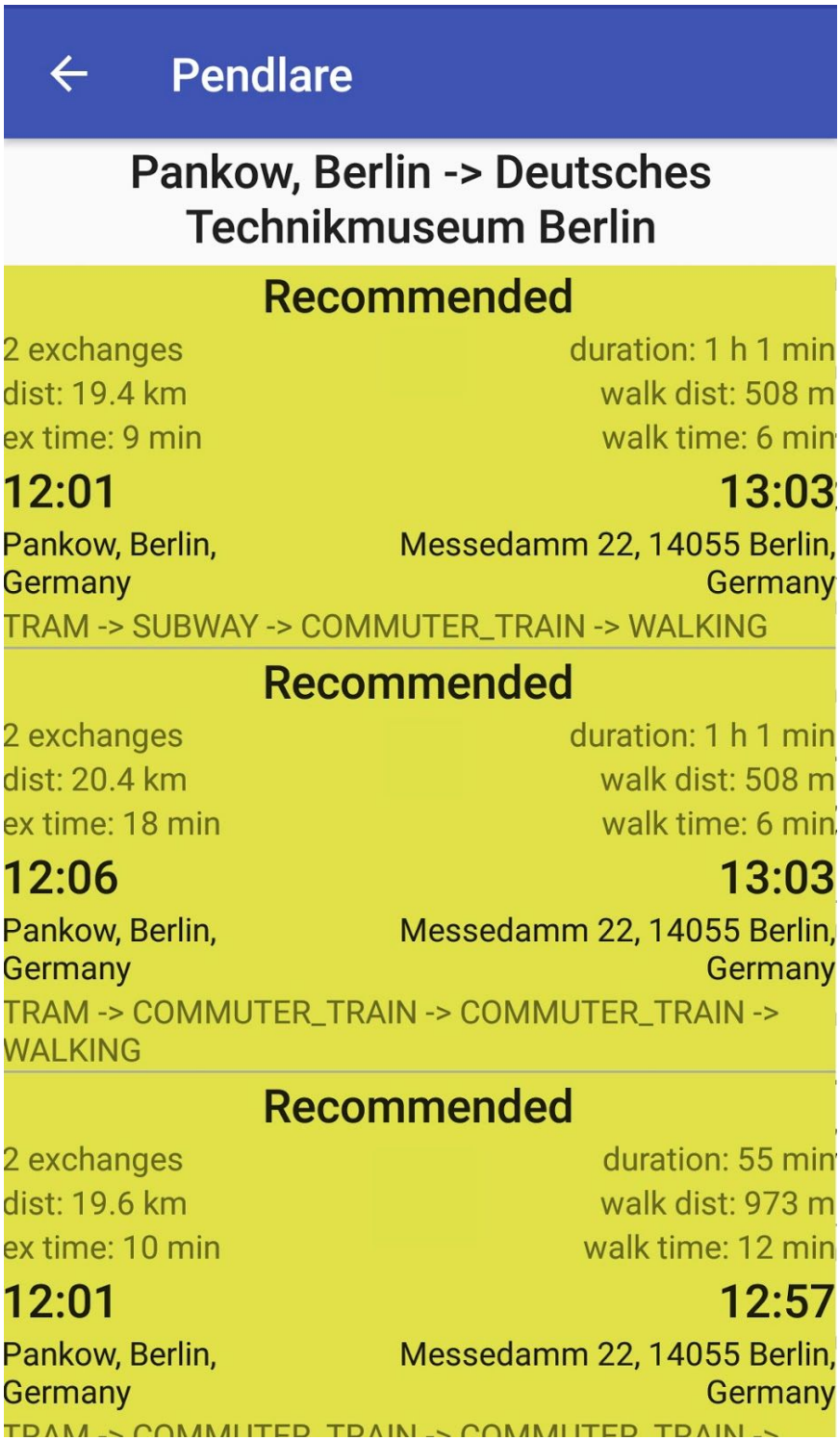
Söksida


Pendlare
SEARCH
From
Pankow, Berlin
To
Messe Berlin
Departure
Arrival
2018-10-01
12:00
Pankow, Berlin
Messe Berlin
Deutsches Technikmuseum Berlin
Reinickendorf, Berlin
Schmargendorf, Berlin

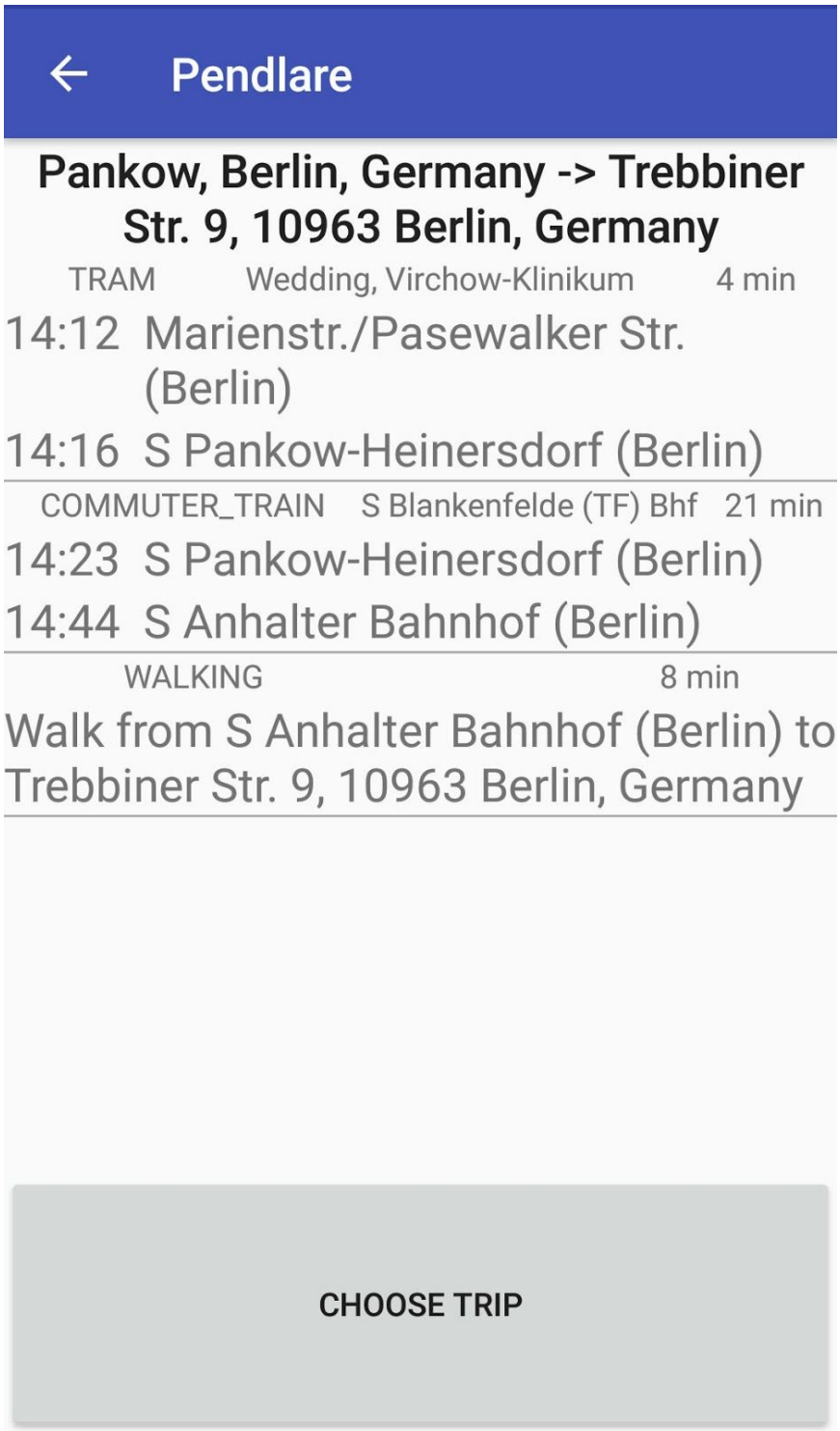
Resultatsida (maskininlärning avstängd)



Resultatsida (Maskininlärning på)



Specifik resultatsida



Printed by Tryckeriet i E-huset, Lund 2019

LUND
UNIVERSITY

Series of Bachelor's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2019-701
http://www.eit.lth.se