Check me out im a 15 year old rapper

Spamklassificering av Soundcloud-kommentarer med Naïve Bayes och SVM

Victor Falini
Viktor Karlberg

HT 19

Kandidatuppsats i statistik, STAH11 (15 hp)
Statistiska institutionen
Lunds universitet
Handledare: Björn Holmquist

Abstract

Soundcloud is the worlds third largest streaming platform for music. Despite this, the website is riddled with spam comments that disturb user experience. This spam differs much in format and content from the email spam that we are used to seeing, and that most classic spam filters have been developed to detect. On soundcloud, 90% of spam comments are written by users wanting to self-promote their own music and contain plenty of slang and spelling errors. Our aim is to develop a machine learning model that can predict spam with a high accuracy for comments on this internet platform. We take several steps to process the data, such as removing stopwords and emojis, in order to facilitate analysis. We use python packages of the Natural Language Processing toolkit in order to process the data and train our algorithms. We test a Naïve Bayes algorithm and two different Support Vector Machines (RBF and Linear kernel) to see which one performs best. Our results show that all three algorithms are still highly effective even when presented with the challenge of detecting spam in this different kind of data; The Support Vector Machine with RBF kernel performs best, predicting 97,3% of comments correctly in our test, but is slower to train than the Naïve Bayes and with only a 2% difference in predictive precision.

Innehållsförteckning

1 Inledning

Soundcloud är den tredje största streamingtjänsten för musik med mer än 300 miljoner registrerade användare och 175 miljoner unika besökare varje månad (Alexa, 2019 ;DMR, 2019). Enligt Alexa, ett företag som rankar alla webbplatser beroende på deras totala webbtrafik, är Soundcloud på 90:e plats i globala rankningen när det gäller webbtrafik och antal besökare (Alexa, 2019). Användare har inte bara möjligheten att lyssna och kommentera på andras låtar, men även möjlighet att ladda upp sina egna låtar. Detta har bidragit till plattformens popularitet där det finns möjligheten för många amatörartister att dela deras musik, helt gratis (Gizmodo, 2014). Under 2014 laddades det upp 12 timmar av ny musik varje minut (DMR, 2019). Trots plattformens storlek så finns det stora problem med spamkommentarer som utgör (data från november 2019) mer än 85 procent av alla kommentarer på topp 50-listan för raplåtar. I en artikel från *The Verge* hävdas det att Soundclouds hjälpsidor har haft en stor mängd användare som klagar på spam-bottar. I samma artikel menar dock Soundcloud att de endast kan ta bort kommentarer som manuellt anmäls av användare och att det inte finns något automatiserat system för hantering av spam (The Verge, 2017). Vår idé blev därför att utveckla ett system för att automatiskt identifiera spam specifikt för kommentarer på Soundcloud, då dessa skiljer sig från den mer klassiska skräpposten som det redan har utvecklats många filter för. Vi förväntar oss att klassificering av vår typ av data kommer att vara svårare eftersom kommentarerna vanligtvis är korta och innehåller mycket slang och stavfel. Vi kommer att använda två olika maskininlärnings-modeller för att göra våra prediktioner; *Naïve Bayes (NB)* och *Support Vector Machine (SVM)*. Vi valde dessa två algoritmer då de i tidigare studier har visat sig vara de mest framgångsrika inom spamklassificering (Dada et al. 2019 ; Manning, Raghavan & Schütze, 2009). Hädanefter kommer vi för enkelhetens skull att benämna icke-spam kommentarer som ham, något som är praxis i litteraturen om ämnet. Spam definieras enligt Nationalencyklopedin som "[...]massutskick på internet av meddelanden som mottagaren vid fritt val skulle avstått från att erhålla, till exempel reklam." (2020). Vi kommer att endast titta på spam för låtar som tillhör amerikanska topp-50 listan för rapp och hip-hop. Valet beror på att det är den mest populära genren och på att låtarna innehåller flest spam-kommentarer och därmed mest data.

2 Data

Data har samlats in genom att använda sig av en webscraper[1] för att automatiskt gå igenom amerikanska topp 50-listan för rapplåtar och ladda ner innehåll och användarnamn för 37000

[1] www.webscraper.io

av kommentarerna. Av dessa 37000 tog vi bort 13000 dubbletter, det vill säga kommentarer som förekommer mer än en gång. Dessa togs bort så endast en upplaga av varje kommentar återstår. Vi tog också bort alla kommentarer som var skrivna av samma användare, eftersom en stor del av spammare kommenterade samma kommentar men med små variationer (exempelvis byta ut ordet "track" med ordet "song"). Dessa kommentarer är inte exakta duplikat men är tillräckligt lika (det är vanligtvis bara ett ord som skiljer kommentarerna) för att det skall finnas en risk för överanpassning av modellen. Detta tillvägagångssätt rekommenderas av Feng, Junwei och Ning (2019). Efter borttagningen av dessa kommentarer hade vi cirka 5400 unika kommentarer kvar. Sedan kodas varje observation med antingen en nolla (ham) eller en etta (spam). Vi har identifierat huvudsakligen tre typer av spamkommentarer:

- "*Self-promotion*" - Den här typen av kommentar har som syfte att öka spelningar, likes och följare av personen som skrivit den. Vanligtvis är kommentaren skriven av Soundcloud-artister som använder sig av kommentarsfältet på populära låtar för att locka andra användare att lyssna på deras låtar. Kommentarerna som tillhör den här kategorin är vanligtvis en variant av "I'm a XX year old rapper, check out my music". Majoriteten av alla kommentarer som vi samlade in tillhör den här kategorin. Vi väljer att klassa Self-promotion kommentarer som spam eftersom de stämmer överens med Nationalencyklopedins definition; samma kommentar återkommer ofta flera gånger per låt och en väldigt stor mängd användare klagar på dessa kommentarer vilket gör att andra kravet också uppfylls. YouTube väljer också att klassificera self-promotion kommentarer som spam vilket vi ser som ytterligare stöd åt vårt klassificerings-val (Google, 2020).

- "*Försäljning*" - Försäljning av diverse produkter är vanligt i kommentarsfälten. Framförallt förekommer försäljning av illegala droger som cannabis, lsd, psykadelisk svamp och MDMA. I vissa fall säljs även licenser för programvaror som FL Studio[2].

- "*Bedrägeri*" - Kommentarer med länkar som utlovar jobb som med väldigt lite arbete och utan någon erfarenhet kan ge mycket pengar.

Efter borttagning av duplikat och klassificeringen av observationer återstår 1097 kommentarer vilka vi har klassat som spam och 4280 som vi har klassat som ham.

Ett urval av fyra typiska spamkommentarer som förekommer är:

> "*This may Go unnoticed, but Im a 17 year Old Rapper And Song Writer from Gulfport*

[2]Ett program för att skapa musik

Mississippi, and If you could listen to My content and Give me any feedback, it would be a Real honor and privilege"

"*you should check out [...]! she makes bedroom pop that's rly cool. ive got a playlist of her stuff, go ahead and listen :)*"

"*We sale all kinds of marijuana products and carts add my snap if interested*"

"*Simple online occupation from home Earns upto $550 to $750 every day by working simply on the web. I have made $28K in this month by working on the web. [...]*"

Nedan följer också ett urval av fyra typiska hamkommentarer:

"*Tripple red is the best*"

"*to much autotune but whatever it is still a nice song*"

"*This is my favorite song*"

"*Wtf the whole comment section is self promote*"

3 Metod

Vi kommer att använda oss av en metod som heter Natural Language Processing, hädanefter benämnt NLP. NLP används i många olika fält, som bland annat spamidentifiering (Medium, 2018). Vi använder oss av programmeringsspråket Python för att den innehåller en stor mängd NLP-verktyg som underlättar databearbetning och analys. För information om koden, se Appendix B. Vi har delat upp metoden i fyra steg:

1. Databearbetning (Bland annat tokenisering, borttagande av skiljetecken och emojis, samt åtgärder mot slang och stavfel).
2. Uppdelning av data i testdel och utvärderingsdel.
3. Träning, Error Analysis och utvärdering av utvalda algoritmer (Naïve Bayes och Support Vector Machine).
4. Analys av resultat.

De fyra stegen gås igenom i detalj på kommande sidor.

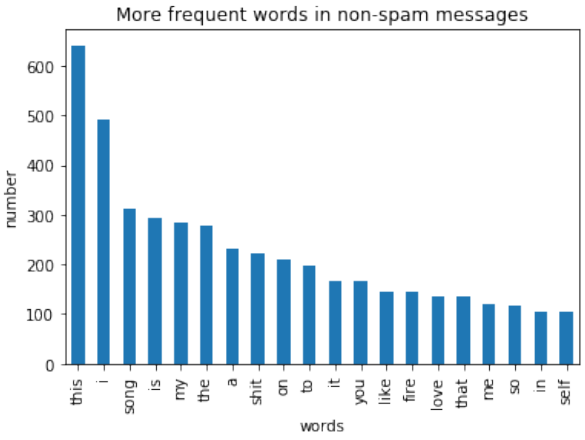
3.1 Databearbetning

3.1.1 Tokenisering

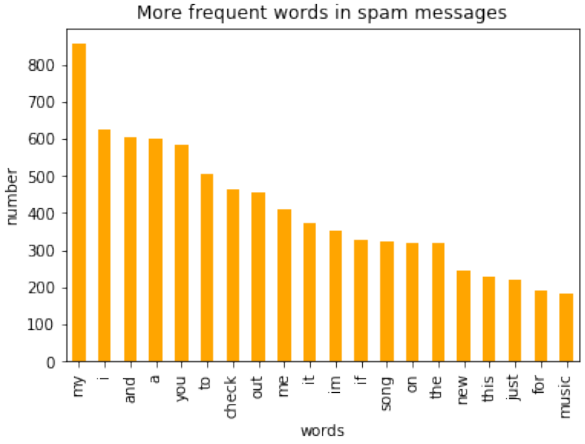
Inom NLP delar man upp all text i det minsta möjliga enheten för analys, så kallade ”tokens” dvs ord (Manning, Raghavan & Schütze 2009). Sedan blir varje unikt ord en diskret variabel där värdet är lika med antal gånger ordet uppstår i kommentaren; Exempelvis kommer meningen ”The pen pen is blue” ha värdet två för variabeln ”pen” och värdet ett för variabeln ”blue”.

3.1.2 Stoppord och skiljetecken

Inom natural language processing tas så kallade stoppord bort då de inte innehåller värdefull information. Exempel på stoppord är ”and”, ”in”, ”you”, ”that”, etc. Samma sak gäller för skiljetecken då dessa inte heller innehåller värdefull information för språkanalys. Borttagning av stoppord rekommenderas av Bramer (2007). Vi använde oss av nltks[3] lista av stoppord med några modifieringar; vi valde att **inte** ta bort stopporden ”my”, ”me” samt ”im” eftersom vi märkte att dessa ord uppkommer med mycket större frekvens i spam- jämfört med ham-kommentarer. Detta går att observera från nedanstående figurer.



Figur 1: Mest använda ord i ham-kommentarer



Figur 2: Mest använda ord i spamkommentarer

Beslutet beror också på att vissa kommentarer endast innehåller tre ord och då vill vi inte förlora information när det redan finns så lite av den.

[3]Listan över engelska stoppord går att hitta på www.nltk.org/nltk_data

3.1.3 Gemener och versaler

Vi har modifierat vår data så att algoritmen inte särskiljer gemener och versaler, detta för att göra det enklare för modellen att förstå att samma ord betyder samma sak även om det är skriven i bara gemener eller bara versaler. Innan den här bearbetningen klassades exempelvis ”FIRE” och ”fire” som två olika ord när faktiska informationen som orden innehåller är likadan i båda fallen. Efter modifieringen är alla ord med enbart gemener.

3.1.4 Slang och stavfel

Då slang används flitigt bestämde vi oss för att ändra på vår data så att ord som har samma betydelse kommer att betraktas som samma variabel av våra algoritmer. Ett exempel på detta är att ”ig” gjordes om till ”instagram” och att ”hmu” gjordes om till ”hit me up”. I vår slutliga modell har vi gjort detta för 29 olika slangord. Då stavfel förekommer mycket i kommentarerna gjorde vi också så att ett ord där samma bokstav förekommer mer än två gånger i rad, förkortas ner till att samma bokstav endast kan förekomma två gånger i rad, ty inget ord i det engelska språket har samma bokstav mer än två gånger i följd. Till exempel blev en kommentar som bestod av ”hellllo” endast ”hello”, detta gjordes för att vår algoritm så lite som möjligt ska bli lurad av felstavade ord.

3.1.5 Emojis

Då både spam- och ham-kommentarer innehåller många emojis, tog vi bort dessa för att reducera det totala antalet variabler i modellen. Detta gjordes även för att vår algoritm hade svårt att särskilja olika emojis från varandra.

3.1.6 Andra språk

Det är inte möjligt för oss att besluta ifall kommentarer skrivna på spanska, ryska och arabiska är spam därför har vi beslutat att ta bort dessa. Då kommentarer på engelska utgör merparten av alla kommentarer anser vi att borttagningen inte leder till några systematiska fel.

3.1.7 N-grams och Stemming

I textklassificering kan det finnas ett behov att slå ihop kombinationer av ord som kan vara betydelsefulla. Till exempel blev en kommentar som bestod av ”my music” transformerad till ”mymusic”, detta för att förtydliga för algoritmen att de två orden tillsammans har en

betydelse. Vi ändrade även böjningen på vissa ord, exempelvis blev ordet ”checking” till dess grundform ”check”. Den här proceduren fyller samma syfte som ovanstående.

3.1.8 Unicode Symboler

En begränsad del av spamkommentarerna använder sig av Unicode Symboler som liknar vanliga bokstäver, men kommer uppfattas som unika bokstäver av vår programvara (IG Fonts Generator, 2019). Detta är ett problem då det för tillfället inte finns något sätt att ”översätta” texten till vanliga bokstäver. Då vi inte vill förlora data har vi manuellt översatt dessa texter till vanliga bokstäver.

3.2 Uppdelning i träning och testdata

Vi delade upp datamaterialet slumpmässigt i två olika delar. Då testdelen används för att träna vår modell vill vi att den ska innehålla så mycket data som möjligt för att få den bästa algoritmen. Därför har vi 80% av observationerna i träningsdelen och på resterande 20% utvärderas vår modell. Då vi har mycket data från början anser vi att utvärderingsdelen, som utgörs av 1076 kommentarer, är tillräckligt stor för att ge tillförlitliga resultat.

3.3 Naïve Bayes

Naïve bayes är en maskininlärningsalgoritm som använder sannolikhetslära och Bayes sats för att klassificera data. Algoritmen har i praktiken visat sig vara mycket effektiv, även med väldigt lite träningsdata och den används mycket inom Natural Language Processing (Bramer, 2007; Rusland et al. 2017). Den är speciellt populär för spamfilter på grund av dess enkelhet och effektivitet (Dada et al. 2019). Algoritmen anses naiv då den bygger på starka antaganden om oberoende mellan X-variablerna i modellen, ett krav som sällan uppfylls i praktiken (Bramer, 2007). Vi utnyttjar Bayes sats:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \tag{1}$$

Ett praktiskt exempel:
Vi vill räkna ut sannolikheten att kommentaren är spam givet att det står ”check out my page” och jämföra det med sannolikheten att kommentaren inte är spam givet att det står ”check out my page”. Sedan klassas kommentaren beroende på vilken av dessa som har största sannolikhet.

$$P(\text{spam}|\text{check out my page}) = \frac{P(\text{check out my page}|\text{spam}) \times P(\text{spam})}{P(\text{check out my page})} \tag{2}$$

$$P(\text{inte spam}|\text{check out my page}) = \frac{P(\text{check out my page}|\text{inte spam}) \times P(\text{inte spam})}{P(\text{check out my page})} \tag{3}$$

Först och främst kan vi ignorera nämnaren i ekvationerna då vi endast är intresserade av vilken av dessa sannolikheter som blir störst för att kunna göra vår prediktion.
Då det inte finns några andra kommentarer som innehåller den här exakta sekvensen av ord så kan vi inte räkna ut sannolikheterna. Men då vi har antagit att sannolikheterna för dessa enskilda ord är oberoende av varandra så kan vi dela upp sannolikheten på följande sätt:

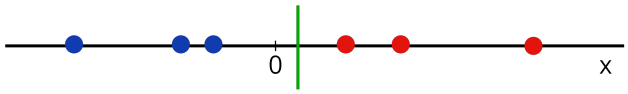
$$\begin{aligned} P(\text{spam}|\text{check out my page}) = P(\text{check}|\text{spam}) \times P(\text{out}|\text{spam}) \times \\ P(\text{my}|\text{spam}) \times P(\text{page}|\text{spam}) \times P(\text{spam}) \end{aligned} \tag{4}$$

Problemet med den här metoden är att vi kan få empiriska sannolikheter som är lika med noll (exempelvis om det finns noll kommentarer med ordet check bland vår kategori inte spam). Dessa kommer att leda till att hela sannolikheten blir lika med noll och därmed förlorar vi all information i resten av vår bayesianska ekvation (Manning, Raghavan & Schütze, 2009). Lösningen till problemet blir att tillämpa en teknik kallad för *Laplace Smoothing* där en konstant (vanligtvis 1) adderas till täljaren för varje sannolikhet så att ingen sannolikhet blir noll (Manning, Raghavan & Schütze, 2009).

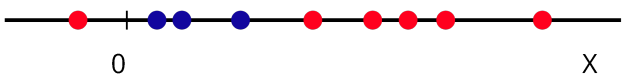
3.4 Support Vector Machine

En support vector machine är en algoritm för att optimalt separera två klasser av data. SVMs bygger på att finna det hyperplan som bäst kan separera data i två klasser. Support Vectors är namnet som man ger punkterna som är närmast planet som skiljer kategorierna och är väldigt viktiga då det är dessa som har störst påverkan på positionen av planet (Hastie, Friedman & Tibshirani, 2009). I figur 3 på nästa sida ser vi en situation där det går att hitta en optimal linje som kan separera våra klasser. Men vad gör man i situationen där det inte går att hitta någon linje som kan separera klasserna, som i figur 4? Lösningen till problemet ligger i att expandera SVM-modellen för att projicera data till en högre nivå, som sen används som en extra dimension för att hitta en ny linje eller plan som kan separera klasserna. Ett exempel på detta visas i figur 5, där vi har tagit vår X-variabel och gjort en lämplig transformation för att hitta en ny linje som kan separera grupperna. Detta kallas för "*the kernel trick*" (Manning, Raghavan & Schütze, s.331 2009). Vi har valt att använda

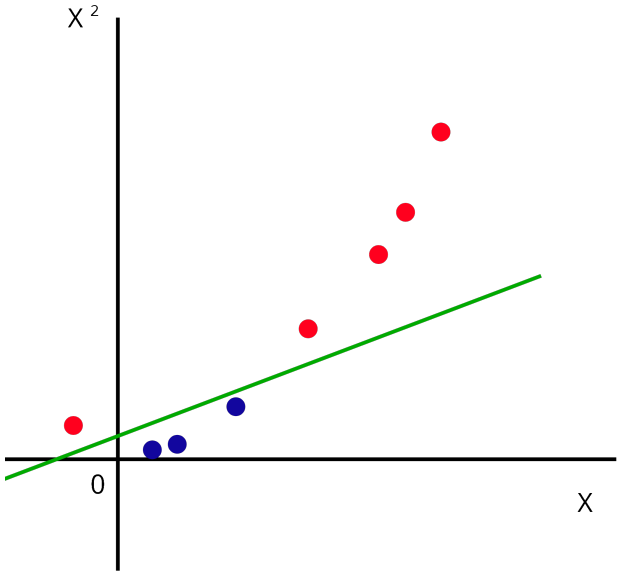
oss av både en linjär och en rbf kernel. Detta eftersom både linjär och rbf kernel har i empiriska undersökningar gett bra resultat (Manning, Raghavan & Schütze 2009 ; Dumais et al. 1998 ; Joachims 1998). Rbf kerneln utnyttjar normalfördelningen för att transformera data till en högre dimension för att på så sätt kunna hitta en linje som avgränsar de två grupperna.



Figur 3: Ett exempel på grupper som går att separera med en linjär gräns

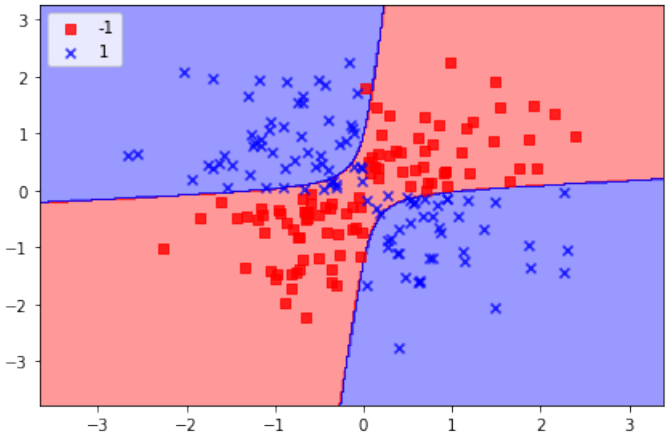


Figur 4: Ett exempel på grupper som inte går att separera med en linjär gräns

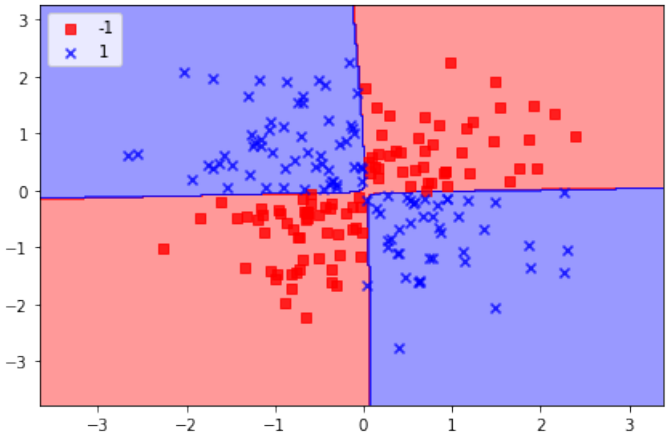


Figur 5: Applicering av en kvadratisk transformation av x-variabeln för att grupperna ska kunna separeras av en linjär gräns

Support vector machine använder sig av en kostnadsparameter, C. Den här parametern kommer att påverka var avskärmningslinjen lägger sig och därmed påverka hur väl modellen klassificerar spam. Stora värden på C innebär att modellen lägger större vikt på de punkter som är nära linjen, det vill säga de punkter som modellen har svårast att klassificera som spam eller inte spam. Mindre värden på C lägger däremot större vikt på de punkter som är längre ifrån linjen. Ett stort värde på C kan leda till en överanpassning av modellen. (Hastie, Tibshirani & Friedman, 2009). Figur 6 och 7 nedan är ett exempel på hur C-parametern påverkar modellen.



Figur 6: Exempel på en SVM (RBF) med C=700. Klassificeringskurvan försöker att separera de två grupperna så bra som möjligt men misslyckas med att klassificera observationerna i mitten rätt



Figur 7: Exempel på en SVM (RBF) med C=10000. Klassificeringskurvan lyckas att separera observationerna i mitten korrekt, men blir sannolikt överanpassad till träningsdata då modellen blir mer komplex

För att hitta det optimala värdet på C rekommenderar Hastie, Tibshirani & Friedman (2009) att man utför en korsvalidering som är tänkt att minimera risken för överanpassning. SVM har visat sig vara som mest effektiv när den undersöker få observationer då dess tillvägagångssätt är komplext. Detta kan leda till en tidskrävande process då datamängden och antalet variabler är stor (Manning, Raghavan & Schütze 2009). För mer information om SVM se Manning, Raghavan & Schütze (2009).

4 Analys och resultat

4.1 Felanalys

Efter att ha testkört algoritmen genomfördes en felanalys[4] i tre steg:

1. Identifiering av felklassificerade kommentarer.
2. Försöka förstå varför elementen felklassificeras.
3. Ändra modellen så den inte blir lurad av fel som är enkla att åtgärda.

Vår första felanalys visar att 35% av felklassificerade kommentarer innehåller stavfel. Detta är troligtvis något som gör det mycket svårare för algoritmen att identifiera spam. Se nedan för två exempel av detta.

fallowe at instagram (pred=ham, actual=spam)

im a 13 year old rappee (pred=ham, actual=spam)

Felanalysen visar att algoritmerna har svårast med att klassificera spamkommentarer av den första typen: "*Self-promotion*" (Se avsnitt 2). Inga av de felklassificerade kommentarerna tillhör de andra två kategorierna av spam. Felanalysen gjordes flera gånger med olika uppdelningar av test och träningsdata och samma resultat erhölls. Resultatet kan tänkas bero både på slumpen och på att den första typen av spamkommentar är vanligast, men då felanalysen genomfördes flera gånger med olika uppdelningar av träning och testdata är detta osannolikt. Vår hypotes är att resultatet beror på att de andra kategorierna av spam, "*Försäljning*" och "*Bedrägeri*", sällan innehåller stavfel och slang. En annan möjlig förklaring är att kommentarer som tillhör dessa andra kategorier ofta innehåller flera ord än kommentarer som tillhör kategorin "Self-promotion". Algoritmerna verkar i allmänhet ha svårt med att klassificera kommentarer som är korta. Detta är inte märkvärdigt då kommentarerna kommer

[4]Översättning av författarna av "error analysis"

att innehålla mindre information. Vi funderade på att introducera ett python-paket för att automatiskt korrigera stavfel men då kommentarerna innehåller mycket slang är det inte möjligt.

Felanalysen tyder även på att algoritmen har svårt att detektera kombinationer av ord som tillsammans är starka tecken på spam. Ett exempel är orden "my" "music" "way" "better". I vanliga kommentarer finns det en sannolikhet att dessa ord förekommer men inte i den här särskilda ordningen. Därför försöker vi hjälpa modellen med så kallade n-grams, som diskuterades i sektion 3.1.7. Vi märkte också att algoritmen hade svårt att identifiera länkar (något som oftast är spam), eftersom varje länk är unik och därför kommer identifieras som ett unikt ord. För att lösa problemet transformerade vi originaldata så att varje ord som innehåller .com kommer att identifieras som en länk. Båda dessa ändringar förbättrade vår modell och tar bort 5 av de felklassifierade observationerna (utav 23 totalt).

4.2 False Positive och False Negative

För att kunna analysera modellernas resultat måste vi först identifiera vilka sorters fel som kan göras i klassificeringen. Vi kan göra två olika typer av fel:

- Ham klassificeras som spam (Vi tar bort kommentarer som inte är spam) : Kallas också för *False Positive*
- Spam klassificeras som ham (Vi släpper igenom spam-kommentarer) : Kallas också för *False Negative*

Vi kan ändra på parametrarna i modellen för att förändra på balansen mellan dessa två felen. För att hitta den optimala balansen vill man fundera kring vilket fel är att föredra. Vanligtvis vid filtrering av mejl vill man hellre släppa igenom spam än att radera viktiga mejl av misstag. Dada et. al skriver: *"When a spam message is wrongly classified as ham, it gives rise to a somewhat insignificant problem, because the only thing the user need to do is to delete such message. In contrast, when a non-spam message is wrongly labeled as Spam, this is obnoxious, because it indicates the possibility of losing valuable information as a result of the filter's classification error"* (2019, s.7). Detta medför att parametrar justeras för att minimera första typen av fel (False Positive). Vi har också valt att optimera våra parametrar för att minimera den här typen av fel.

4.3 Resultat

Vi testar modellerna med 200 möjliga värden på parametrarna (laplace smoothing-parametern och C-värdet) för att hitta kombinationen som minimerar fel av den första typen. Resultaten visar på att bästa modellen i vårt fall är en SVM(RBF) med ett C-värde på 700. Den här modellen lyckas att klassifiera 97,3% av testdata korrekt. Vi korsvaliderade modellen för att visa att detta värde på C är det optimala och att den inte beror på slumpmässiga uppdelningen av test- och träningsdata. Linjära modellen för SVM presterar också bra i prediktionen av spam, och gör detta för ett C-värde på ett. Naïve Bayes är den som presterar sämst av alla modeller, men har en tydlig fördel i hastighet då man kan träna modellen på mindre än en sekund medan SVM modellerna tar mycket längre tid att träna. Nedan presenteras resultaten av klassificeringen av testdata för våra tre modeller. Notera att även fast Naïve Bayes modellen presterar sämst, så kommer alla tre modeller ha en likartad prediktiv förmåga. För SVM-modellerna kommer tiden som algoritmen tar att öka exponentiellt när antal variabler ökar (Dada et al. 2019).

Naïve Bayes (Laplace smoothing=1)

	Predicted Ham	Predicted Spam
Actual Ham	830	33
Actual Spam	17	196

Tabell 1: *Confusion matrix* för Naïve Bayes, Accuracy= 95.3%

SVM (Linear, C=1)

	Predicted Ham	Predicted Spam
Actual Ham	845	18
Actual Spam	23	190

Tabell 2: *Confusion matrix* för linjär SVM , Accuracy= 97.2%

SVM (Radial Basis Function, C=700)		
	Predicted Ham	Predicted Spam
Actual Ham	855	8
Actual Spam	21	192

Tabell 3: *Confusion matrix* för RBF SVM , Accuracy= 97.3%

Vi har använt oss av några python-verktyg för att analysera hur modellerna tar beslut. Då SVM med rbf inte är linjär kan den inte analyseras av dessa verktyg och därför kan det vara svårt att få insyn i hur algoritmen tar sina beslut. I Tabell 4 presenteras de 15 mest inflytelserika orden för Naïve Bayes-algoritmens beslutsfattande. Bland dessa ord finns stopporden som vi valde att inte ta bort i vår data-bearbetningsdel; me, im och my (under formen my music). Notera att när loglikelihoodvärdet närmar sig noll så kommer vikten av ordet i klassificeringen att vara större. I tabell 5 presenteras de 15 mest inflytesrika orden för vår Linjära SVM som visar att modellen verkar lägga stor vikt på vissa specifika ord även om dessa förekommer endast en gång i spamkommentarerna (två exempel på detta är orden frog och corndog som troligtvis inte är bra prediktorer på spam). Vi ser detta som en form av överanpassning till träningsdata men trots detta visar våra resultat att SVM-modellen presterar mycket bra i praktiken. Resultatet är lite märkligt då ett litet C-värde borde, teoretiskt sett, minimera risken för överanpassning (se sektion 3.4).

Loglikelihood	Ord
-3.9114	me
-4.0795	song
-4.1227	im
-4.4352	new
-4.5694	check
-4.6471	check out
-4.7517	listen
-4.8306	like
-4.9663	follow
-4.9749	pls
-5.0100	my music
-5.1134	self
-5.2287	would
-5.2741	give
-5.2741	called

Tabell 4: Loglikelihoodvärden för spam givet ett visst ord. Värdena är negativa eftersom det är logaritmerade sannolikheter.

Vikt	Ord
+4.881	im way better
+3.999	check me out
+3.999	sharetap2earncokavronb
+3.999	fwm
+3.999	repost
+3.999	im better
+3.999	follow
+3.352	frog
+3.308	promotion
+3.111	com
+2.972	checkout
+2.745	beats
+2.624	making
+2.494	corndogs
-2.999	<BIAS>(Intercept)

Tabell 5: Vikter för linjära SVM prediktorn. En positiv vikt medför att en kommentar som innehåller ordet har en ökad sannolikheten för att vara spam.

5 Diskussion och slutsats

I den här uppsatsen har vi skapat en algoritm som ska detektera spamkommentarer på Soundcloud. Vi använde oss av tre olika modeller som gav olika bra resultat. Naïve Bayes-modellen är den vars tillvägagångssätt är enklast att förstå då den bygger på vedertagen statistik och grundar sig på Bayes sats. Support vector machine, som de andra två modellerna grundar sig på, är svårare att förstå men de båda visade sig vara bättre än Bayes-modellen på att detektera spam. Våra resultat stämmer överens med tidigare studier som visar på att Naïve Bayes är en metod som, trots dess teoretiska brister, fungerar bra i praktiken.
Resultaten visar att SVM-algoritmen med rbf kernel (C=700) presterar bäst då den klassificerar 97,3% av testdata korrekt. Linjära SVM-algoritmen presterar lite sämre med 97.2% och Naïve Bayes presterar sämst med 95,3%. Vår studie ger stöd åt tidigare forskningsresultat som visar att både Naïve Bayes och SVM-algoritmer är mycket bra på spamklassifi-

cering. Vår felanalys visar att en stor begränsning i klassificering av vår data är slang och felstavade ord då 35% av felklassificerade kommentarer innehåller stavfel och många andra innehåller slang. Felstavade ord är inte den enda begränsningen då det också finns problem med Unicode-Symboler som inte uppfattas som vanlig text av programvaran (se sektion 3.1.8). Vår felanalys visar också att algoritmerna har lättare att identifiera klassiska typer av spam (*Försäljning* och *Bedrägeri*) jämfört med spammen som är specifik till Soundcloud (*Self-promotion*). Vi har också visat att vid klassificering av Soundcloud-kommentarer är det bra att ha kvar stoppord som "my" "me" och "im". Analys av de mest inflytesrika orden visar att vår Linjära SVM verkar överanpassas mer till träningsdata jämfört med vår Naïve Bayes-modell, även vid lågt C-värde. Detta hade troligtvis åtgärdats om mer data samlats in.

Vi hade tyvärr inte möjlighet till att använda modeller med neurala nätverk eller så kallade *rule based classifiers* och kan därför inte jämföra våra resultat för att veta vilken algoritm bland alla dessa som är optimal för klassificering av den här sortens data. Vi hade inte heller möjligheten att mäta och presentera träningstiden för de olika algoritmerna. För framtida studier hade det därför varit intressant att se hur andra sorters algoritmer och modeller presterar på Soundcloud-data samt studera hur tidseffektiva de är jämfört med våra modeller.

6 Referenser

Alexa. (2019). Tillgänglig online: https://www.alexa.com/siteinfo/soundcloud.com [Hämtad 2 december 2019].

Bramer, M. (2007). Principles of data mining, London: Springer.

Dada, E., Bassi, J., Chiroma, H., Abdulhamid, S., Adetunmbi, A. & Ajibuwa, O. (2019). Machine learning for email spam filtering: review, approaches and open research problems, *Heliyon*, vol. 5, no.6.

DMR. (2019). 16 amazing Soundcloud statistics and facts,
Tillgänglig online: https://expandedramblings.com/index.php/soundcloud-statistics/ [Hämtad 3 december 2019].

Dumais, S., Platt, J., Hecherman, D., & Sahami, M. (1998). Inductive Learning Algorithms and Representations for Text Categorization. *Proceedings of the seventh international conference on Information and knowledge management.* s. 148-155.

Feng, H., Junwei, W. & Ning, B. (2019). Spam message classification based on the Naïve Bayes classification algorithm, *IAENG International Journal of Computer Science*, vol. 46, no. 1, s. 46-53.

Gizmodo. (2014). How Soundcloud changed music forever, Tillgänglig online: https://gizmodo.com/how-soundcloud-changed-music-forever-1588811594 [Hämtad 1 december 2019].

Google. (2020). Tillgänglig online: https://support.google.com/youtube/answer/2801973?hl=en-GBref_topic=9282365 [Hämtad 15 januari 2020].

Hastie, T., Tibshirani, R. & Friedman, J. (2009). The elements of statistical learning, 2a uppl., New York: Springer.

IG fonts generator. (2019). Tillgänglig online: https://igfonts.io/ [Hämtad 19 december 2019].

Joachims, T. (1998). Text categorization with Support Vector Machines: Learning with

many relevant features, in: Nédellec C., Rouveirol C. (eds) *Machine Learning: ECML-98. ECML 1998. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence)*, s. 37-42, Berlin: Springer.

Manning, C., Raghavan, P. & Schütze, H. (2009). Introduction to information retrieval, Cambridge: Cambridge university press.

Medium. (2018). Introduction to natural language processing: Part 1, Tillgänglig online: https://medium.com/analytics-vidhya/introduction-to-natural-language-processing-part-1-777f972cc7b3 [Hämtad 16 december 2019].

Nationalencyklopedin. (2020).
Tillgänglig online: https://www.ne.se/uppslagsverk/encyklopedi/lång/spam Hämtad [17 januari 2020].

Rusland, N., Wahid, N., Kasim, S. & Hafit, H. (2017). Analysis of Naïve Bayes algorithm for email spam filtering across multiple datasets, *IOP conference series: Materials science and engineering*, vol. 226, no. 1.

The Verge. (2017). How Soundcloud's broken business model drove artists away, Tillgänglig online: https://www.theverge.com/2017/7/21/15999172/soundcloud-business-model-future-spotify-streaming [Hämtad 22 november 2019].

7 Appendix A

NB - Naïve Bayes
SVM - Support Vector Machine
RBF - Radial Basis Function
NLP - Natural Language Processing
Ham - Icke-spam

8 Appendix B

```
# Importera paket

import pandas as pd
import numpy as np
import pandas as pd
import nltk
import math
from nltk.corpus import stopwords
import string
import functools
import io

# Importera data från excel

df = pd.read_excel(io.BytesIO(uploaded['soundcloudtop50.xlsx']))
df.columns = ['spam','usr','comment']
df = df[pd.notnull(df['spam'])]

# Importera stoppord

stop_words = set(stopwords.words('english'))
not_stopwords = 'me', 'my'
final_stop_words = set([word for word in stop_words if word not in not_stopwords])

# Borttagning av emojis

df['comment'] = df['comment'].str.replace('[^#@/:%.,_-]', '')

# Fixa länkar

df['comment'] = df['comment'].str.replace('.com', ' .com ')

# Gör alla bokstäver till gemener
```

a =df.iloc[:,2]
b = np.array(a).astype(str)
c = np.char.lower(b)

Definierar funktion som genomför tokenisering samt bearbetar vissa stavfel. Tar även bort en bokstav om samma bokstav förekommer fler än två gånger i följd.

def process_text(text):
cleantext = ".join(cleantext)
cleantext = functools.reduce(lambda x,y: x+y if x[-2:]!=y*2 else x, cleantext, "")
#fix slang
cleantext = cleantext.replace("sending","sendin")
cleantext = cleantext.replace("amosc","amos")
cleantext = cleantext.replace("follows", "follow")
cleantext = cleantext.replace("youtube", "yt")
(Fullständig lista inte inkluderad p.g.a längd)
finaltext = [word for word in cleantext.split() if word.lower() not in final_stop_words
return finaltext

Splittar upp data i träning- och utvärderingsdel

from sklearn.model_selection import train_test_split x_train, x_test, y_train,
y_test = train_test_split(commentbag, df['spam'], test_size = 0.2, random_state = 1)

Träning av Naïve Bayes modell

from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
classifier.fit(X_train, y_train)

#Utvärdering av Naïve Bayes

from sklearn.metrics import classification_report,confusion_matrix, accuracy_score
pred = classifier.predict(X_test)
print(classification_report(y_test ,pred))
print('Confusion Matrix: ', confusion_matrix(y_test,pred))

print('Accuracy: ', accuracy_score(y_test,pred))

Liknande metod användes för SVM-modellerna, därför inte inkluderad.