# 1 Introduktion

Inom sentimentanalys studeras och analyseras individers åsikter, sentiment, attityder och känslor gentemot olika entiteter som produkter, tjänster, organisationer, problem, andra individer och ämnen. En av de mer välkända uppgifterna inom området är sentimentklassificering som definieras som den automatiserade processen för identifiering och klassificering av sentiment uttryckta i text. På så sätt försöker man bestämma en ”författares” attityd gentemot ett ämne, produkt eller situation, baserat på de åsikter som uttrycks i texten. Sentimentklassificering har många applikationer, till exempel inom e-handel, där företag ges möjligheten att snabbt kunna analysera kunders feedback och recensioner av en tjänst eller produkt och på så sätt kunna anpassa sin verksamhet till sina kunders behov och åsikter.

Innan tillkomsten av Deep Learning användes statistiska metoder och maskininlärningstekniker för sentimentanalys och klassificering. Med ökade datamängder, förbättrad tillgänglighet till textkorpus, ökad beräkningsstyrka och förbättrade algoritmer har neurala nätverk allt mer kommit att användas. Detta har lett till markant förbättrade resultat inom olika NLP uppgifter, bland annat inom sentimentklassificering. Med denna bakgrund har vår uppgift varit att, givet en text innehållande en serie- eller filmrecension, predicera dess sentiment utifrån 5 olika klasser med Naiv Bayes och Neurala nätverk för att sedan evaluera vilken metod som presterar bäst.

# 2 Bakgrund

## 2.1 Balansering av träningsdata

Problem med obalanserad träningsdata uppstår när skillnaden är stor i mängden data mellan klasser. Allmänt gäller att klassificerare tränade på obalanserad data tenderar att bli överväldigade av de vanliga klasserna och därför blir dåliga på att predicera de underrepresenterade klasserna. En möjlig lösning är att öka kostnaden för missklassificering av den/de underrepresenterade klasserna i relation till de överrepresenterade klasserna. Idén med en kostnadsfunktion är att om vi tänker oss att en falsk negativ (FN) är värre än en falsk positiv (FP) så kommer vi att pennalisera den FN som exempelvis 10 FP. Detta betyder att en FN för den underrepresenterade klassen nu kostar lika mycket som 10 FN för den överrepresenterade klassen.

Oversampling och undersampling är andra vanliga lösningar. Oversampling innebär att man duplicerar datapunkter från den/de underrepresenterade klasserna så att de har mer effekt på algoritmen medan undersampling innebär att man tar bort datapunkter från den/de vanliga klasserna så att de får mindre effekt på algoritmen. Av dessa lösningar är undersampling den vanligaste. En potentiell nackdel med undersampling är dock att metoden tar bort variation från den/de vanliga klasserna vilket innebär en ökad risk för overfitting för berörda klasser jämfört med övriga balanseringmetoder. Samtidigt är denna effekt liten om man arbetar med stora datamängder.

## 2.2 TF–IDF

TF-IDF står för Term-Frequency–Inverse Document Frequency. Metoden vektoriserar och mäter hur relevant en term är inom ett textdokument relativt till vår kollektion av textdokument, det vill säga vårt korpus av recensioner. TF-IDF värdet för ett ord fås av produkten av ordets Term Frequency och Inverse Document Frequency (Kap 6, Jurafsky & Martin, 2023). Term Frequency är antalet gånger ett givet ord förekommer i en recension relativt antalet ord i hela recensionen medan Inverse Document Frequency är ett mått på hur mycket information ordet ger, det vill säga om det är vanligt eller sällsynt i alla dokument (i vårt fall recensioner). På så sätt skalar IDF ner vikten av ord som förekommer i väldigt många recensioner eftersom de är mindre informativa än ord som bara förekommer i en mindre andel av vårt korpus. Ord som är relativt unika och bara förekommer i en liten andel av våra recensioner ges då högre relevans än ord som är gemensamma för merparten av våra recensioner som exempelvis “movie”, “film”, eller “like”. Ett ords relevans är högre när ordet förekommer mycket i en given recension men väldigt sällan i andra recensioner, så vanligheten av ordet i en recension mätt som Term Frequency blir balanserad med sällsyntheten mellan recensioner mätt som Inverse Document Frequency.

## 2.3 Naiv Bayes

Multinomial Naive Bayes implementerar Naiv Bayes algoritmen på multinomialt distribuerad data. Den baseras på Bayes teorem som säger att särdragen i datan är ömsesidigt självständiga, där förekomsten av ett särdrag inte påverkar sannolikheten av förekomsten för andra särdrag. Detta är en av de mer klassiska varianter av Naiv Bayes som används för text- och sentimentklassificering, där datan är representerad antingen som ordvektorer enligt word count vektorer eller som TF–IDF vektorer som nämns längre fram. Distributionen parametriseras genom vektorerna $\theta_k$ för varje klass $k$, vars element är sannolikheten $P(x_i|k)$ att ett visst specifikt särdrag $i$ förekommer i en text som tillhör klass $k$ (Kap 4, Jurafsky & Martin, 2023). Alltså beräknar algoritmen sannolikheten av varje klass, givet ett stickprov av text, och ger sedan klassen med högst sannolikhet som output.

## 2.4 Neurala Nätverk

Ett *feedforward network* är en av de enklare typerna av neurala nätverk. Det består av flera neuronlager (*neuron layers)* där vi har ett indata-lager, gömda lager och ett utdata-lager. Detta tillåter datorer att identifiera mönster och särdrag för att lösa specifika problem, på ett sätt som delvis är tänkt att reflektera beteendet av den mänskliga hjärnan. Man kan tänka sig varje artificiellt neuron som en egen linjär regressionsmodell där vikterna av kopplingen mellan neuronerna avgör hur mycket inflytande datapunkt i indatan har på utdatan.

I *feedforward* nätverk är neuronerna kopplade utan någon cykel. Med det menas att utdatan från varje neuron i ett lager blir indatan till neuronerna i nästa lager, utan att någon utdata skickas tillbaka till föregående lager som redan processats (Kap 7, Jurafsky & Martin, 2023).

Dock kan man skapa en arkitektur där detta tillåts vilket nämns senare i rapporten (se sektion 7.1.4). Målet är att minimera kostnadsfunktionen för att få en så bra anpassning (*fit*) som möjligt för varje given observation, vilket sker genom gradient descent när modellen justerar sina vikter för att anpassa träningsdatan. Gradient descent bygger i sin tur på att modellen minimerar en given kostnadsfunktionen. Man behöver inte programmera nätverket explicit för att det ska lära sig utan upplärningen hanteras av nätverket självt likt den mänskliga hjärnan.

# 3 Beskrivning av implementation

## 3.1 Beskrivning och rensning av data

Totalt har vi utgått från 1 010 293 recensioner från en större datamängd av drygt 5,5 miljoner recensioner från Internet Movie Database (IMDb). Datan är hämtad från Kaggle (2021) som är en tävlingsplattform inom Data Science och onlinecommunity för maskininlärningsutövare och datavetare. Den hämtade datan består bland annat av en detaljerad recension, kallad ”review_detail”, av den berörda filmen/serien och ett betyg, ”rating”, av filmen/serien på en skala 1-10. Den detaljerade recensionen, hädanefter refererad till enbart som recension, är den text som rensats och därefter använts som input för klassificeringen medan betyget omvandlats till våra klasser.

Av tidigare nämnda 1 010 293 recensioner har enbart 958 773 kunnat användas, då resterande 51 520 recensioner saknar recensionsbetyg. Som avgränsning har dessutom alla recensioner kortare än 200 tecken tagits bort, vilket ger totalt 711 236 kvarvarande recensioner. Anledningen till denna avgränsning är att kortare recensioner gör det svårare att predicera rätt klass eftersom våra modeller inte har tillgång till lika mycket data. Som beskrivs senare i implementationsdelen kan också nämnas att många ord försvinner vid borttagning av stoppord samt till följd av att våra modeller, i varierande utsträckning, bara tar hänsyn till de vanligaste orden. För en betydande del av recensionerna med mindre än 200 ord hade därför våra modeller endast haft ett fåtal ord, och ibland inga ord alls, att göra prediktioner utifrån.

I den nedladdade data ges recensionsbetyg på en skala 1-10. Som förenkling har antalet klasser dock minskats från 5 till 10 enligt tabell 1. Detta motiveras av att det med tio klasser är ganska godtyckligt om en person exempelvis sätter en nia eller tia i betyg.

| Ursprunglig klassificering | 1 & 2 | 3 & 4 | 5 & 6 | 7 & 8 | 9 & 10 |
|---|---|---|---|---|---|
| Ny klassificering | 1 | 2 | 3 | 4 | 5 |

*Tabell 1. Omdefiniering av klasser för recensionsbetyg.*

## 3.2 Balansering av träningsdata

Av de 711 236 recensioner som använts har 601 236 använts som träningsdata. Dessa har balanserats genom undersampling enligt tabell 2, genom att behålla samtliga recensioner från klass 2 och slumpmässigt välja 61 939 recensioner från var och en av de övriga klasserna.

| Klass | 1 | 2 | 3 | 4 | 5 | Totalt |
|---|---|---|---|---|---|---|
| Träningsdata före balansering | 85 819 | 61 939 | 97 285 | 158 973 | 197 220 | 601 236 |
| Träningsdata efter balansering | 61 939 | 61 939 | 61 939 | 61 939 | 61 939 | 309 695 |

*Tabell 2. Fördelning av träningsdata mellan klasser före och efter balansering.*

## 3.3 Rensning av recensionstext

Recensionstexten har rensats i två steg. Först har vi på egen hand skrivit och använt reguljäruttryck för att ta bort alla tecken utom bokstäver A-Z, a-z och mellanslag. Detta har i sin tur gjorts i flera steg, så att till exempel två eller fler punkter i rad först ersatts med mellanslag varefter två eller fler mellanslag i rad ersatts av ett mellanslag. På sätt som detta har vi försökt kompensera för att de som ursprungligen skrev in recensionen ibland glömt mellanslag vid början av en ny mening. Efter att alla reguljäruttryck applicerats har stora bokstäver ersatts med små.

I steg två, har biblioteket spaCY använts för att lemmatisering och borttagning av stoppord. Totalt inkluderar spaCY 326 stoppord som bland annat "a", ”an”, ”about”, ”after”, ”put”, ”show” och så vidare. Dessutom har ord kortare än tre tecken tagits bort, vilket dock inte gör så stor skillnad efter att alla stoppord tagits bort.

## 3.4 Vektorisering av recensionstext

För att kunna göra prediktioner utifrån vår rensade recensionstext måste vi göra om dessa till något som en textklassificeringsmodell kan förstå. Det är en ganska generell process att transformera en kollektion av textdokument till numeriska särdragsvektorer. Vi använder oss av en av de mer populära metoderna inom NLP och informationshämtning, nämligen TF–IDF. Specifikt har Scikit Learns vektoriserare *TfidfVectorizer* (Scikit Learn, 2023b) använts för att skapa en vokabulär med de 10 000 vanligaste orden från vår balanserade data. Modulens *transform*-metod har därefter använts för att transformera vår tränings- och testdata till TF–IDF vektoriserade versioner, för att göra om varje recension till en vektor med 10 000 element motsvarande TD-IDF värdena för de 10 000 ordens unika index.

## 3.5 Naiv Bayes

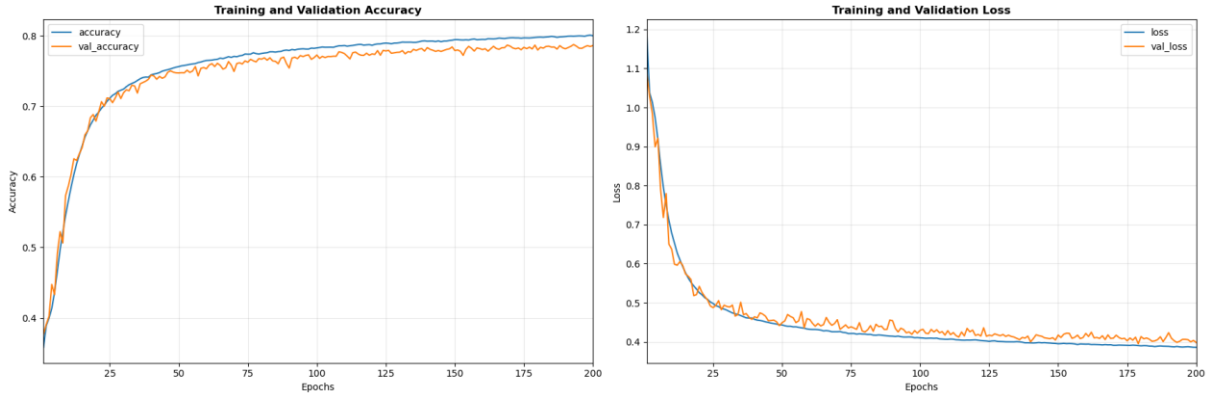
För Naive Bayes har Scikit Learns klassificerare *MultionmialNB* (Scikit learn,2023a) använts. Först har en *fit* gjorts på vår träningsdata och därefter har dess *predict*-metod använts på vår testdata för att predicera vilken klass recensionerna i testdatan tillhör. Detta har gjorts både för vår balanserade och obalanserade data, för att illustrera skillnaden i prestation. Notera att implementationen i bägge fallen gjorts utan att optimera hyperaparameterar eller alfa-värden som representerar Laplace-smoothing parametern i detta fall.

## 3.6 Neurala nätverk

Som input till det neurala nätverket har samma TD-IDF vektorer använts som för Naiv Bayes, med skillnaden att vektorerna kortats ner från 10 000 till 1 000 element motsvarande de 1000 viktigaste orden. Vilka av de 10 000 vanligaste orden som är de 1 000 viktigaste har bestämts utifrån TFIDF-vektoriserarens sannolikhetsvektorer som fås genom attributet feature_log_prob_. Specifikt ger feature_log_prob_ för ett enskilt ord en vektor med log-sannolikheterna för våra fem klasser. De 1 000 viktigaste orden har bestämts genom att välja de ord, eller i själva verket ordindex, som har största skillnad mellan max- och minvärdet för sina vektorelement. Till exempel finns för ett ord som ”unwatchable” en stor skillnad i log-sannolikheter mellan olika klasser, varför detta ord tagits med, medan ett ord som ”movie” har en mycket låg skillnad i log-sannolikheter mellan klasser och därför exkluderats. Försök har även gjort med att öka längden av inputvektorerna till att inkludera de 2500 viktigaste orden, dock utan märkbar förbättring i prediktion.

Själva nätverket har konstruerats med Tensorflow och utformats med fyra gömda lager med 1000 neuroner vardera. Rectified Linear Unit (ReLU) har använts som aktiveringsfunktion för samtliga gömda lager. När det neurala nätverket konstruerades användes först endast två gömda lager, vilket ökades till tre eftersom detta gav en bättre en modell. Jämförelsevis sågs en förhållandevis blygsam förbättring när antalet gömda lager ökades till fyra, varför det inte ansetts motiverat att använda mer än fyra lager.

Istället för att implementera one-hot encoding och predicera en sannolikhet som summerar till 1 för de fem klasserna har valet gjort att implementera klassificeringen med regression, alltså med ett outputlager med en enda neuron som predicerar den slutliga klassen. Anledningen till detta val är att vi med regression kan använda medelkvadratfel för förlusten, vilket gör att en femma som felklassificeras som en etta pennaliseras betydligt hårdare än femma som felklassificeras som en fyra. Specifikt har outputlagret försetts med en egenkonstruerad aktiveringsfunktion som liknar ReLU, men som ger ett tal mellan 0,5 och 5,5. Den predicerade klassen är detta tal avrundat till närmsta heltal. Modellen har tränats genom att köra igenom all träningsdatan 200 gånger enligt figur 1. Trots avsaknaden av regularisering ger modellen hela tiden en ökad noggrannhet och minskad förlust, dock med en något ökande diskrepans mellan tränings- och testdata.




*Figur 1. Noggrannhet och förlust (medelkvadratfel) för validering och träningsdata för neuralt nätverk.*

# 4. Metod för testning och evaluering

För testning och evaluering har 10 000 slumpmässigt utvalda recensioner använts. Till skillnad från träningsdatan, har testdatan inte balanserats eftersom testdatan är tänkt att spegla hur den faktiska betygsfördelningen ser ut. Observera att obalanserad testdata alltså använts oavsett om en modell tränats på balanserad eller obalanserad data.

Vid utformningen av de modeller som presenteras i resultatsdelen har också samma mätvärden som presenteras i resultatsdelen använts för testning och evaluering, det vill säga precision, täckning, F1-score och noggrannhet. I fallet med de tre först mätvärdena är det framförallt det viktade medelvärdet som använts eftersom det lätt blir oöversiktligt att hålla reda på dessa värden separat för samtliga fem klasser. Dessutom har förväxlingsmatriser likt dem som visas i figur 2 använts flitigt för att visuellt, översiktligt och i detalj, evaluera våra modeller. En förväxlingsmatris ger nämligen all information som går att få om hur en modell klassificeras data. Den kod som behövs för att kunna visualisera klassificeringsfördelningen i en förväxlingsmatris utvecklades därför redan i ett tidigt skede under projektet.

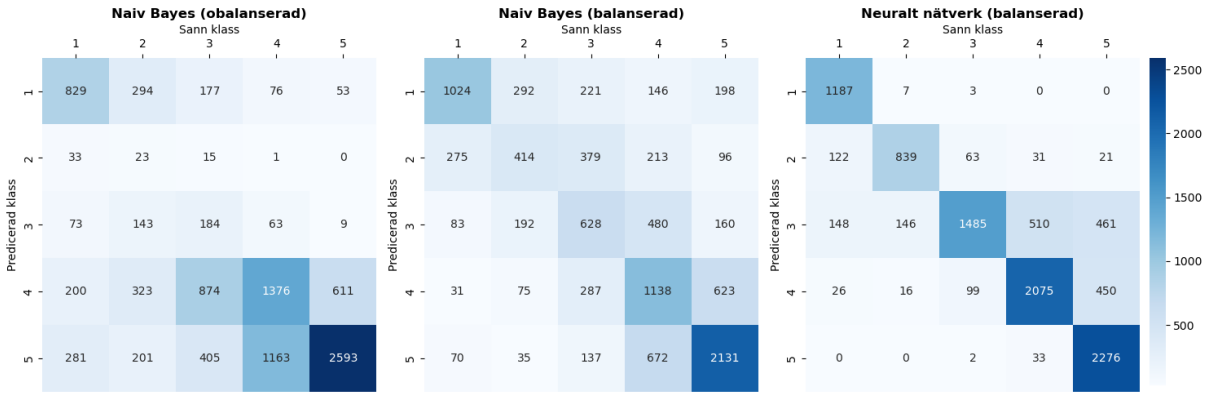
Vad gäller det neurala nätverket har dock mest vikt lagts vid att minimera förlustfunktionen, medelkvadratfelet. Anledningen är dels att det är just medelkvadratfelet modellen försöker minimera, inte noggrannheten, precisionen, täckningen eller F1-värdet, även om dessa värden indirekt förbättras ju lägre förlusten blir. För det andra, eftersom noggrannheten, precisionen, täckningen och F1-värdet indirekt förbättras vid en minskande förlust är det egentligen inte motiverat att hålla reda på dessa värden separat. Dock behövs fortfarande en viss koll så att exempelvis precisionen inte blir bättre på bekostnad av täckningen eller tvärtom.

# 5. Resultat och Analys

Om vi börjar med att jämföra resultaten för Naiv Bayes ger balanserad data en måttlig förbättring av samtliga fyra evalueringsvärden. Exempelvis ger Naiv Bayes tränad på balanserad data en noggrannhet på 53,3% medan noggrannheten för Naiv Bayes tränad på obalanserad data är 50,0 %. I figur 2 ses också visuellt en tydligt skillnad i det att det framträder en relativt tydligt diagonal för den version av Naiv Bayes som tränats på balanserad data. Specifikt framgår att antalet sanna positiver är högre för klass 1-4 och framförallt för klass 2 och 3 som är de mest underrepresenterade klasserna. Inte helt oväntat är dock antalet sanna positiver för den vanligaste klassen, klass 5, sämre med balanserad data.

En ännu tydligare förbättring fås av det neurala nätverket som har en noggrannheten på 78,6%. Som framgår av figur 2 och tabell 3 fås betydligt bättre resultat för samtliga evalueringsvärden. Visuellt ger det neurala nätverket helt klart den tydligast framträdande diagonalen, alltså flest sanna positiver. Det är också värt att notera att det neurala nätverket inte klassificerar någon etta som en femma eller femma som en etta, medan motsvarande siffror är 70 respektive 198 för den version av Naiv Bayes som tränats på balanserad data. En viktig anledning till detta tros vara att medelkvadratfelet kunnat användas som förlustfunktion för det neurala nätverket eftersom det baseras på regression. Samtidigt bör påpekas att

precisionen för klass 3 är förhållandevis låg, 54,0%, vilket visserligen är betydligt bättre än för Naiv Bayes men mer än 20 procentenheter sämre än för övriga klasser. Anledningen till detta skulle emellertid också kunna vara att medelkvadratfelet använts som förlustfunktion, eftersom den i genomsnitt pennaliserar felklassificerade treor minst.




*Figur 2. Förväxlingsmatriser för evaluering av Naiv Bayes och neuralt nätverk på obalanserad testdata. Den första versionen av Naiv Bayes har tränats på obalanserad data medan den andra versionen av Naiv Bayes och det neurala nätverket tränats på balanserad data.*

| | **Naiv Bayes** | | | **Neuralt nätverk** | | | |
|---|---|---|---|---|---|---|---|
| Klass | Precision | Täckning | F1 | Precision | Täckning | F1 | Support |
| 1 | 54,4 | 69,0 | 60,9 | 99,2 | 80,0 | 88,6 | 1 483 |
| 2 | 30,1 | 41,1 | 34,7 | 78,0 | 83,2 | 80,5 | 1 008 |
| 3 | 40,7 | 38,0 | 39,3 | 54,0 | 89,9 | 67,5 | 1 652 |
| 4 | 52,8 | 43,0 | 47,4 | 77,8 | 78,3 | 78,1 | 2 649 |
| 5 | 70,0 | 66,4 | 68,2 | 98,5 | 70,9 | 82,5 | 3 208 |
| Medel | 54,3 | 53,3 | 53,4 | 83,7 | 78,6 | 79,5 | 10 000 |

*Tabell 3. Precision, täckning och F1-score (i procent) för Naiv Bayes och neuralt nätverk tränad på balanserad data. Tabellen motsvarar delfigur 2 och 3 i figur 2. Medel avser viktad medelvärde.*

# 6 Diskussion

En intressant frågeställning utifrån resultatet är varför det neurala nätverket ger så mycket bättre resultat än Naiv Bayes. Inputen till de bägge modellerna är nämligen exakt densamma bortsett från att det neurala nätverket enbart ges information om förekomsten av de 1 000 viktigaste orden av de 10 000 vanligaste som Naiv Bayes ges som input. En avgörande skillnad är dock att det neurala nätverket inte bara betraktar förekomsten av varje ord för sig utan kombinationer av förekomsten av två eller fler ord. Till exempel, kanske en recension som både innehåller orden ”disappointed” och ”surprised” bör klassificeras annorlunda än genomsnittet för två recensioner som var för sig enbart innehåller det ena ordet. Denna typ av korrelation och beroende mellan inputvariabler kan det neurala nätverket lära sig och ta hänsyn till. Anledningen är att det neurala nätverket består av flera lager, i vårt fall fyra.

Den teoretiska bakgrunden varför MNB ger sämre resultat är antagandet om naiv självständigheten mellan varje par av särdrag, det vill säga att alla inputvariabler är stokastiskt självständiga varandra. Om antagandet inte är uppfyllt betyder det att det existerar korrelation mellan inputvariabler som kommer att ha en negativ effekt på modellens noggrannhet. I vårt fall är detta villkor inte uppfyllt, vilket gör Naive Bayes till en suboptimal modell.

Vad gäller uppgiftens svårighet kan också kommenteras att det är betydligt svårare att gradera sentiment på en 5-gradig skala, från negativt till positivt, jämfört med ett generellt klassificeringsproblem med fem klasser. Exempelvis skulle författarattribution, genre- eller ämnesklassificering med fem distinkta klasser troligtvis gett oss betydligt bättre resultat. Särdragen mellan klasserna skulle i detta fall förmodligen varit mer explicita och tydliga än med våra recensioner. Till exempel inom ämnesklassificering av artiklar kommer modellen titta på om orden i inputsekvensen, som handlar om exempelvis datorer, är nära associerade med modellens uppfattning om specifika särdrag om ämnet datorer som modellen lärt sig. Detta gör det lättare för modellen att inse att en sekvens som handlar om fotboll inte innehåller element som grafikprocessorer eller RAM. I vårt fall är det inte så explicit och tydligt vad som särskiljer en tvåa från en trea eller en fyra från en femma eftersom de inte är lika unika som för artiklar.

För att delvis komma runt detta problem, att våra klasser innehåller så få distinkta särdrag, har det neurala nätverket implementerats med regression och medelkvadratfel som förlustfunktion. Som kommenteras i resultatsdelen gör detta att det neurala nätverket inte klassificerar en enda etta som en femma eller femma som en etta, medan motsvarande siffror är 70 respektive 198 för Naiv Bayes (totalt 10 000 datapunkter). Denna möjlighet att implementera klassificering med regression hade vi inte haft, till exempel, i fallet med artikelklassificering om vi haft fem distinkta klasser som fotboll, datorer, politik, ekonomi och hälsa. Man kan inte säga att det är bättre att felklassificera en artikel som handlar om fotboll som en ekonomiartikeln än som en datorartikel. Däremot är det utan tvekan bättre att felklassificera en recensionen av klass 5 som en fyra än som en etta. Just beroendet mellan klasserna är något vi utnyttjat och sannolikt en bidragande orsak varför resultaten för det neurala nätverket är så mycket bättre än för Naiv Bayes.

Samtidigt är det, som påpekas i resultatsdelen, troligt att kombinationen av regression och medelkvadratfel som förlustfunktion också förklarar varför precisionen för klass 3 för det neurala nätverket är mer än 20 procentenheter lägre än för övriga klasser. Så som det neurala nätverket är implementerat pennaliseras nämligen felklassificerade treor minst i genomsnitt.

# 7 Slutsats

Resultaten i denna rapport visar tydligt att ett neuralt nätverk, som är en mer avancerad modell, ger betydligt bättre resultat än en enklare modell som Naiv Bayes. Vi har också visat att det indirekt är möjligt att kombinera neurala nätverk med Naiv Bayes, för avgöra vilka ord som är tillräckligt betydelsefulla och därmed minska storleken på det neurala nätverket utan att nämnvärt göra avkall på dess prediktionsförmåga. Vad gäller prediktionsförmågan hos det

neurala nätverket bör dock understrykas att ett neuralt nätverk, till skillnad från Naiv Bayes, inte på samma sätt är en färdig modell som bara direkt kan implementeras. För att utforma ett neuralt nätverk med en bra prediktionsförmåga krävs inte obetydliga kunskaper. Det är därför troligt att vi med bättre kunskaper kunnat utforma ett liknande neuralt nätverk med betydligt högre prediktionsförmåga. I delkapitlet som följer går vi igenom några av de förbättringsområden som identifierats.

Dessutom har vi visat på vikten av att använda balanserad träningsdata, inte bara för att bättre kunna predicera de underrepresenterade klasserna, men också för att ge en bättre prediktionsförmåga för modellen som helhet. När vi balanserade vår träningsdata förbättrades nämligen vår modells förmåga att predicera de underpresenterade klasser avsevärt medan förmågan att predicerade vår vanligast klass sjönk betydligt mindre.

## 7.1 Förbättringsområden

### 7.1.1 Inkludering av fler särdrag

En relativt enkel möjlig förbättring skulle kunna vara att lägga till den ursprungliga recensionslängden som ett nytt särdrag. I vår data kan vi se att recensioner i snitt är längre för de mer neutrala klasserna 2-4, varför detta särdrag skulle kunna användas för att bättre särskilja dessa klasser från övriga. Ett annan möjlig förbättring skulle kunna vara att lägga till antalet utropstecken som ett särdrag, då utropstecken i den nuvarande implementationen tas bort med reguljäruttryck innan lemmatiseringen. I vår data kan vi se att utropstecken är upp till dubbelt så vanliga i klass 1 och 5 som i klass 3.

### 7.1.2 Borttagning av stoppord

Det finns både för- och nackdelar med att ta bort stoppord från våra recensioner. Framförallt finns många fördelar kopplade till att vi genom att minska textstorleken gör att datan kan hanteras på ett mindre krångligare och snabbare sätt. Detta kan förbättra prestationen av NLP algoritmer genom att reducera antalet icke-relevanta ord som algoritmen behöver hantera samt förbättra tolkningen av resultatet. Samtidigt finns nackdelar kopplade till risken att förlora relevant information när ord tas bort som kan vara signifikanta i en specifik kontext. Ett exempel i vårt fall är “He was not a good actor” som skulle bli “he”, “good”, “actor” efter tokenisering och borttagning av stoppord, vilket är det motsatta till den ursprungliga betydelsen.

Därför måste man vara försiktigt med vad exakt som tas bort och vilka konsekvenser det får beroende på vilken metod man använder. Vid användning av metoder som TF–IDF eller count vectorization tas visserligen ingen hänsyn till kontexten och därför finns stora fördelarar med att ta bort en stor andel stoppord eftersom det minskar dimensionella rummet. Dock borde en stor förekomst av ett stoppord som ”not” rimligtvis bidra till negativt sentiment och det vore därför rimligt att undanta vissa stoppord. Ett annat exempel är ordet ”ever” som kan antas vara vanligare för både de mest positiva och de mest negativa recensionerna. Därför föreslår vi en villkorad borttagning av stoppord baserad på vårt innehåll där mer

betydelsefulla stoppord som “not” och “ever” undantas medan resterande ord som “a”, “the”, ”an”, “and” och ”about” tas bort precis som tidigare.

### 7.1.3 Villkorad lemmatisering

Vi föreslår även en villkorad lemmatisering av den rensade texten där enbart verb men inte adjektiv lemmatiseras. Detta för att förhindra att laddade ord transformeras ner. Ett exempel i vårt fall är att adjektivsformerna “good”, “better” och “best” alla lemmatiseras till “good” med nuvarande implementationen, vilket påverkar styrkan av det positiva sentimentet. Detta skulle leda till att en recension tillhörande klass 5 felklassificeras till en lägre klass. Däremot bör verb fortfarande lemmatiseras eftersom det inte finns någon motsvarande skillnad i sentiment mellan exempelvis “watch”, “watches”, “watched” och “watching”.

### 7.1.4 Neurala Nätverk som tar hänsyn till kontext

Recurrent Neural Networks (RNN) är speciellt anpassade till sekventiella data (sekvenser av ord i vårt fall). Till skillnad från mer allmänna feed-forward neurala nätverk, använder inte RNN en hel sekvens i ett svep utan RNN processar sekvensen element för element, där nätverket i varje steg inkorporerar nya data med den information nätverket redan har arbetat med. Detta är ganska likt hur vi människor tar in sekvenser, nämligen att vi läser en mening ord för ord där vi i varje steg tar in och tyder ett nytt ord för att sedan inkorporera det med betydelsen av de ord vi redan läst (Kap 9.1, Jurafsky & Martin, 2023).

Även om vanliga RNN kan hantera varierande längder av sekvenser så lider RNN av det så kallade “vanishing gradient problem”, nämligen att gradienter under bakåtpasset (backpropagation) blir extremt små vilket medför en ökad svårighet för nätverket att lära sig av datan. För att minimera detta problem kan man använda sig av en Long-Short Term Memory (LSTM) metod som tar bort information som inte behövs från kontexten, och lägger till information som nätverket tror kommer behövas för senare beslut (Kap 9.5, Jurafsky & Martin, 2023).

Ett tillägg till LSTM modellen är att även göra den bidirektionell (BiLSTM). För uppgifter likt sekvensklassificering med RNN så används nätverkets sista gömda tillstånd som indata till ett senare framåtpassklassificerare. En svårighet med denna arkitektur är att det sista tillståndet reflekterar mer information om slutet av sekvensen än sekvensens början. Genom att använda en bidirektionell arkitektur (Kap 9.4, Jurafsky & Martin, 2023) så tillåter vi nätverket att använda information från både den föregångna och framtida kontexten för sina prediktioner. Detta görs genom att konkatenera det sista gömda tillståndets framåt- och bakåtpass och därefter använda det som indata i nästa steg. Vi testade att implementera detta för det binära fallet vilket gav bra resultat. Vi tror därför att det med rätt nätverksarkitektur och optimering av hyperparameter är det möjligt att åstadkomma en betydlig ytterligare förbättring av resultaten jämfört med det neurala nätverk vi implementerade.

# 7 Referenser

Haibo, He & Yunqian, Ma., 2013. "*Imbalanced Learning Foundations, Algorithms, and Applications*". Wiley, 2013

Jurafsky, Dan & Martin, James H., 2023. "*Speech and Language Processing*" (3rd ed. draft.) Stanford Education, 2023. https://web.stanford.edu/~jurafsky/slp3/

Kaggle, 2021. *IMDb Review Dataset - ebD*. https://www.kaggle.com/datasets/ebiswas/imdb-review-dataset Hämtad 15 Dec 2023

*MultionmialNB*, 2023a. Modul från programmeringsbibliotek Scikit-Learn. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html Hämtad 15 Dec 2023

*TfidfVecorizer*, 2023b. Modul från programmeringsbibliotek Scikit-Learn. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html#sklearn.naive_bayes.MultinomialNB Hämtad 15 Dec 2023