Finalklar

Vilka bidrag kvalificerar sig för ESC finalen – ett ML projekt



Matilda Wilhelmsson

EC Utbildning

Kunskapskontroll projektkurs

202410

# Abstract

This project aims to create a machine learning model capable of predicting whether a contestant in Eurovision song contest (ESC) semi final will qualify for the final competition or not. We use data from Kaggle and Musicstax. We create 8 different models: Decision Tree, Logistic Regression, Random Forest and a Neural Network (and the same ones using Random oversampling). The Random Forest model with random oversampling performed best with a prediction accuracy of 77%.

Innehållsförteckning

[1 Inledning 1](#_Toc181355171)

[2 Teori 2](#_Toc181355172)

[2.1 Klassificeringsmodeller 2](#_Toc181355173)

[2.1.1 Decision Tree 2](#_Toc181355174)

[2.1.2 Logistisk Regression 2](#_Toc181355175)

[2.1.3 Random Forest 2](#_Toc181355176)

[2.2 Neurala Nätverk 3](#_Toc181355177)

[2.2.1 Feedforward Neural Network 3](#_Toc181355178)

[2.3 Random oversampling 3](#_Toc181355179)

[2.4 Transformeringsmetoder 3](#_Toc181355180)

[2.4.1 Label Encoding 3](#_Toc181355181)

[2.4.2 One Hot Encoding 4](#_Toc181355182)

[2.5 Bedömningsmetriker 4](#_Toc181355183)

[2.5.1 Accuracy 4](#_Toc181355184)

[2.5.2 F1-score 4](#_Toc181355185)

[3 Metod 5](#_Toc181355186)

[3.1 Agil arbetsmetodik 5](#_Toc181355187)

[3.2 Datainsamling 5](#_Toc181355188)

[3.3 EDA och variabelselektion 5](#_Toc181355189)

[3.4 Modeller 6](#_Toc181355190)

[3.4.1 Applicera Random Oversampling 6](#_Toc181355191)

[3.4.2 Tränings- och testdata 6](#_Toc181355192)

[3.4.3 Decision Tree 6](#_Toc181355193)

[3.4.4 Logistisk Regression 6](#_Toc181355194)

[3.4.5 Random Forest 7](#_Toc181355195)

[3.4.6 Neuralt nätverk (FNN) 7](#_Toc181355196)

[4 Resultat och Diskussion 9](#_Toc181355197)

[5 Slutsatser 10](#_Toc181355198)

[Källförteckning 11](#_Toc181355199)

# Inledning

Eurovision Song Contest gick av stapeln för första gången 1956 i Schweiz. Endast 7 nationer deltog och de som inte var på plats fick ta del av tävlingen via radio. Efter att över 60 år har passerat så följs nu tävlingen av miljoner världen över. Under ESC 2024 som hölls i Malmö, Sverige så deltog 37 olika nationer. (Eurovision official website)

ESC har blivit något som många samlas kring varje år. De tv-sända programmen ses gärna i sällskap med vänner och familj och bidragen väcker starka känslor, både positiva och negativa. Detta leder naturligt in på diskussioner kring hur bra det kommer gå för olika bidrag i tävlingen.

Olika betting-sidor erbjuder betting just när det kommer till ESC. Det finns en uppsjö av olika bets att lägga men de vanligaste rör just placering och huruvida ett bidrag kommer kvalificera sig för finalen eller inte. Vissa lägger ner mycket till på att läsa in sig på bidragen och själva försöka göra en prediktion. Med hjälp av ML-modeller så kan vi i stället systematisera prediktionerna genom att mata in alla variabler vi tror kan vara relevanta för utfallet och låta modellen spotta tillbaka en prediktion. (Eurovision world)

Vad är det egentligen som gör att ett bidrag kvalificerar sig för finalen? Hur kan vi prediktera rätt? Denna frågeställning är inte endast viktig för de som planerat att satsa pengar på någon bettingsida utan även för den tävlingslyste som gärna vill vinna över sina vänner i tv-soffan.

Syftet med denna rapport är att utveckla olika ML-modeller som kan prediktera huruvida en låt i ESC kommer gå vidare från semi-final till finalen, för att uppfylla syftet så kommer följande frågeställning(ar) att besvaras:

1. Kan vi med hjälp av ML prediktera vilka låtar som går vidare till ESC finalen?

# Teori

För att besvara frågeställningen och uppnå syftet med rapporten behöver vi implementera olika modeller som kan hjälpa oss att prediktera ESC kvalificeringar. I följande avsnitt ges en överblick över de modeller som valts ut för ändamålet.

## Klassificeringsmodeller

Klassificeringsmodeller är ett samlingsbegrepp på de modeller som används för att lösa klassifieringsproblem, problem där vi behöver kunna dela in datapunkter i olika grupper. Modellerna skiljer sig mycket åt i hur de löser klassificeringsproblemen och vilka problem de lämpar sig för.

### Decision Tree

Decision Tree-modellen har fått sitt namn då den bildar en struktur som liknar ett träd vars grenar delar på sig ju längre bort från stammen vi kommer. Vid varje förgrening ställs datapunkterna inför en fråga kring en variabel och beroende på svaret så skapas två eller fler nya grenar. Varje gren representerar en undergrupp av data, och processen upprepas på varje nivå tills beslut tas i form av ett blad i trädet. Modellen är kraftfull vid både klassificerings och regressionsproblem och även enkel att tolka, men kan riskera att överanpassa data om den inte beskärs ordentligt. (Géron, A. 2019)

### Logistisk Regression

Logistisk regression är trots sitt namn en klassificeringsmodell som lämpar sig för klassifieringsproblem menat att prediktera binära utfall (som i vårt fall, ett bidrag klassificerar sig eller ej). Modellen använder en logistisk funktion för att modellera sannolikheten för att en observation tillhör en av två klasser, där utgången vanligtvis är 0 eller 1. Genom att använda en sigmoid-funktion omvandlas linjära kombinationer av indata till en sannolikhetsskattning mellan 0 och 1. Logistisk regression fungerar bäst när det finns ett linjärt samband mellan de oberoende variablerna och log-odds av utfallet. (Géron, A. 2019)

### 

### Random Forest

En random forest modell består av flera olika decision trees. Den är en så kallad ensemble-modell, en modell som bygger på flera modeller som kombineras. De prediktioner som de olika decision trees resulterar i räknas samman och den klass som totalt fått mest prediktioner blir den vinnande klassen. Trots modellens relativa simpelhet är den en av de mest kraftfulla maskininlärningsmodeller som existerar för närvarande. (Géron, A. 2019)

## Neurala Nätverk

Neurala nätverk inspirerades ursprungligen av hjärnan. De är uppbyggda så att olika neuroner skickar information sinsemellan. Modellen är uppbyggd av olika lager av dessa neuroner. Neurala nätverk är väldigt kraftfulla när det kommer till att identifiera komplexa mönster i stora datamängder. (Géron, A. 2019)

### Feedforward Neural Network

Ett Feedforward Neural Network (FNN) är ett typ av neuralt nätverk där signalerna mellan neuronerna endast rör sig i framåtriktning. Detta innebär att information flödar från inmatningslagret, genom ett eller flera dolda lager, och vidare till utmatningslagret utan återkoppling eller slingor. Varje neuron i ett lager är ansluten till alla neuroner i nästa lager, vilket gör FNN till ett "fullständigt anslutet" nätverk. Modellen används ofta för klassificerings- och regressionsuppgifter, där den tränas genom att justera vikterna mellan neuronerna baserat på felmarginalen i modellens förutsägelser. (Géron, A. 2019)

## Random oversampling

Random oversampling är en enkel teknik som innebär att man på ett randomiserat sätt duplicerar datapunkter från minoritetsklassen bland featurevariabeln så att den innehåller lika många datapunkter som majoritetsklassen. Detta är användbart när man jobbar med ett litet dataset där det finns en obalans bland klasserna för featurevariabeln. Modellerna får på så sätt lika många datapunkter att träna de olika utfallen på. Tekniken kan dock leda till överanpassning av modellen så den bör användas varsamt. (Wisam, E. 2023)

## Transformeringsmetoder

Många dataset innehåller både kategoriska och numeriska data. För att kunna träna en maskininlärningsmodell på ett tillfredställande sätt är det dock fördelaktigt att omvandla de kategoriska variablerna till numeriska variabler. Detta kan ske på olika sätt.

### Label Encoding

När man applicerar label encoding på en kategorisk variabel så omvandlas de kategoriska värdena till numeriska värden. Varje kategori får ett eget unikt heltalsvärde. Denna teknik passar särskilt bra när det finns en naturlig rangordning bland de kategoriska variablerna. Om inte en sådan rangordning finns bör man undvika att använda tekniken då modellen ändå kan tolka in ett rangordnat förhållande mellan variablerna. (sci-kit learn)

### One Hot Encoding

När man applicerar One Hot Encoding på en variabel så skapas en ny binär kolumn för varje kategori. Så istället för en kolumn med tre olika värden så skapas tre kolumner där varje rad kan anta 1 eller 0 för kolumnen beroende på vilket kategoriskt värde raden har. Tekniken passar bra för variabler där kategorierna inte har någon rangordning sinsemellan. (Géron, A. 2019)

## Bedömningsmetriker

För att kunna bedöma prestationen hos maskininlärningsmodeller behöver vi hitta värden och metriker som ger oss möjlighet att utvärdera prestationen snabbt och träffsäkert. Nedan följer en introduktion av två av dessa metriker.

### Accuracy

Accuracy visar hur väl en maskininlärningsmodell kan prediktera rätt. Värdet på accuracy varierar mellan 0 och 1 och visar hur stor andel av prediktionerna som var korrekta utav de totala prediktionerna gjorda av modellen. Var 80 utav 100 prediktioner rätt så resulterar detta i ett accuracy-score på 0.8. Denna metrik är värdefull när vi snabbt vill kunna utvärdera hur vår modell har presterat på datan. Man bör vara vaksam om man jobbar med obalanserade dataset där ett utfall är vanligare än de andra eftersom accuracy då kan anta ett högt värde även fast modellen inte predikterar de andra utfallen rätt. (Géron, A. 2019)

### F1-score

För att säkerställa att en maskininlärningsmodell verkligen är bra på att prediktera rätt så bör man komplettera accuracy med F1-score. F1-score är en kombination av en modells recall(sanna identifierade positiva värden) och precision(möjlighet att fånga in alla positiva värden). Själva matematiken bakom F1-score går ut på att skapa ett balanserat medelvärde mellan precision och recall. Denna metrik används vid fördel när man jobbar med obalanserade dataset. (Géron, A. 2019)

# Metod

I detta avsnitt kommer metoden redogöras för stegvis med avstamp i en generell beskrivning av den agila arbetsmetodiken.

## Agil arbetsmetodik

Vi har under hela arbetet haft daglig kontakt via Microsoft Teams och Whatsapp. Vi har även setts ett par gånger varje vecka och arbetat tillsammans. Tack vare regelbunden kontakt så har vi behållit flexibiliteten genom hela projektet när vi stött på oväntade problem. Vår prio har varit enkel men kraftfull kod som levererar det den ska utan att krångla till förloppet.

## Datainsamling

Datan som används i detta projekt kommer från Kaggle.com. Kaggle är en sida som samlar en uppsjö av olika databaser som bl. a. privatpersoner har laddat upp. Vi använder datasetet ”Eurovision Song Contest Data” som har laddats upp av användaren ODI6S. Användaren har i sin tur manuellt samlat in datan från Wikipedia, Youtube och Tunebat(en sida med detaljerad information om olika låtar). Vid EDA så upptäcktes att 80 låtar saknade vissa data. Vi lyckades fylla på manuellt 60 av dessa med data från Musicstax.com. Resten av låtarna togs bort från datasetet. Datasetet från Kaggle innehåller data om eurovision åren 2009–2023.

## EDA och variabelselektion

För att på en smidigt men komplett sätt få överblick över vårt dataset så använde vi oss av python-paketet ydata\_profiling. Paketet innehåller funktionen ProfileReport som genomför en EDA på en pandas DataFrame-object. EDAn visade att datasetet innehöll totalt 565 låtar och 44 variabler. Totalt saknade 2448 celler värden.

Vi började med att göra en variabelsektion med hjälp av ProfileReport. Variabler som saknade värden för alla låtar samt variabler som endast var kopplade till finalen sållades bort då de inte var av någon vikt för vår analys av semi-finalerna. Vissa variabler sållades bort då värdena var extremt ojämnt fördelade.

Efter att vi genomfört selektionen så återstod totalt 17 kolumner och 438 låtar. Vissa kolumner representerade kategoriska värden som behövde One Hot Encodas. Vår key-variabel var rangordnat kategorisk och den applicerade vi Label Encoding på.

Vi valde att separera datan för låtarna under 2023 och använda dessa för att göra prediktioner på med de färdiga modellerna. Dessa sparades i ett separat dataset och target-variabeln raderades.

För att få en överblick över datan efter att vi selekterat och transformerat den så använde vi oss återigen av ProfileReport. Denna visade att vi nu hade 62 variabler och 438 låtar.

## Modeller

### Applicera Random Oversampling

Då vår targetvariabel var lite obalanserad och datasetet är litet så valde vi att skapa en kopia av vårt dataset som vi applicerade Random Oversampling på. Detta resulterade i att vi för båda utfallen fick 257 låtar. För att se om det blir skillnad i modellprestation beroende på om vi balanserat datan eller ej så testar vi alla modeller två gånger. Först på den obalanserade datan och sen på den balanserade.

### Tränings- och testdata

För att kunna träna och testa modellerna så delade vi upp vårt obalanserade och balanserade dataset i tränings- och testdata. Fördelningen vi använde var 85% av data fick agera träningsdata och 15% testdata. Vi använde även funktionen StandardScaler på datan då olika kolumners värden varierade starkt.

### Decision Tree

Först tränade och testade vi en standard DecisionTreeClassifier med den obalanserade datan och tog fram statistik på hur väl den presterade.

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

Vi gjorde sedan samma sak fast med den balanserade datan.

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

### Logistisk Regression

Vi tränade, testade och utvärderade den logistiska modellen med obalanserad data. Vi satte antalet iterationer till 10 000.

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

Vi fortsatte på samma sätt med den balanserade datan.

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

### Random Forest

Vi tränade, testade och utvärderade Random Forest-modellen med obalanserad data. Vi satte antalet iterationer till 10 000.

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

Vi fortsatte på samma sätt med den balanserade datan.

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

### Neuralt nätverk (FNN)

Vi tränade, testade och utvärderade vårt neurala nätverk på den obalanserade datan. Vi använde oss av två gömda Dense-lager med 32 neuroner i vardera. Som aktiveringsfunktioner använde vi *relu* och *sigmoid* och som optimeringsfunktion använde vi *adam.* Batch\_size sattes till 64 och antalet epoker 55.

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

Vi gjorde sedan samma med den balanserade datan.

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

# Resultat och Diskussion

|  |  |  |
| --- | --- | --- |
| **Modellutvärdering** | |  |
| **Modell** | **Accuracy** | **F1-score** |
| Decision Tree | 0,7576 | 0,8095 |
| Decision Tree (Oversampling) | 0,7692 | 0,7805 |
| Logistisk regression | 0,6515 | 0,7294 |
| Logistisk regression (Oversampling) | 0,5641 | 0,5652 |
| Random Forest | 0,6667 | 0,7660 |
| Random Forest (Oversampling) | 0,7692 | 0,7955 |
| Neuralt nätverk (FNN) | 0,6667 | 0,7442 |
| Neuralt nätverk (FNN) (Oversampling) | 0,6923 | 0,6923 |

Tabell 1: Accuracy och F1-score för de åtta olika modellerna.

Den modell som presterade bäst var Random Forest med oversampling. Modellen uppnådde en accuracy på 77% och ett F1-score på 0,796.

Random Forest är en kraftfull typ av ensemble modell som ofta presterar bra i olika sammanhang när den jämförs med andra modeller. Modellen minskar risken för överanpassning då den delar upp datan slumpmässigt i delmängder som de olika beslutsträden får träna på. Modellen presterar bättre med den balanserade datan och vi har ett bättre F1-score. Detta beror antagligen på att med balansering så blir fördelningen av target-variabeln jämnare över de olika beslutsträden. Träningen blir helt enkelt effektivare.

Överlag så ser vi inte någon stor påverkan av att balansera datan med hjälp av Random Oversampling. För den logistiska modellen blev faktisk accuracy sämre när den balanserade datan användes. Detta kan bero på att obalansen inte var så stor i vårt dataset och därför gav inte balanseringen någon drastisk effekt.

Vi fick avvikande resultat när vi använde Decision Trees för att prediktera låtarna från 2023(vårt prediktionsset). Decision trees predikterade sannolikheten binärt, 1 eller 0, för att bidragen kvalificerar sig för finalen. Detta trots att vi använt funktionen predict\_proba som ska ge oss den exakta sannolikheten för prediktionen. Denna avvikelse tyder på att träden är överanpassade. För att motverka detta så skulle vi kunnat begränsa träddjupet och antal samples som krävs för att en gren ska dela sig.

Utifrån vår ProfileReport kunde vi se att inga av våra valda variabler hade särskilt stark korrelation med target-variabeln. Detta påverkar möjligheten att träna träffsäkra modeller. Vid ett fortsatt arbete skulle man kunna prova att ta med nya variabler såsom antal streams en låt har innan framträdande, olika online-undersökningar etc.

# Slutsatser

Ursprunglig frågeställning var: Kan vi med hjälp av ML prediktera vilka låtar som går vidare till ESC finalen?

Det går att träna maskininlärningsmodeller för att förutsäga vilka låtar som går vidare till ESC finalen. Alla modeller vi har tränat i detta projekt är bättre på att prediktera rätt än slumpen. Den bästa modellen, Random Forest med oversampling, hade en accuracy på 77%.

# Källförteckning

Eurovision official website. <https://eurovision.tv/>

Eurovision world. <https://eurovisionworld.com/odds/eurovision-semi-final-1>

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow.* (2. uppl.) Sebastopol: O’Reilly Media, Inc.

Kaggle.com. *Eurovision Song Contest Data*. <https://www.kaggle.com/datasets/diamondsnake/eurovision-song-contest-data/data>

Musicstax. <https://musicstax.com/>

Sci-kit learn. <https://scikit-learn.org/dev/modules/preprocessing.html#preprocessing-categorical-features>

Wisam, E. (2023). *Class Imbalance and Resampling: A Formal Introduction* <https://towardsdatascience.com/class-imbalance-and-oversampling-a-formal-introduction-c77b918e586d>