Gruppe 16 ITMAL Slutprojekt

Jonas Nielsen, *201805353*, *201805353@uni.au.dk*
Bjarke Damsgaard Eriksen, 201708652, *au576289@uni.au.dk*

11.December 2020

Navn / Ansvarsområde	Bjarke Eriksen	Jonas Nielsen
Rapport		
Introduktion	P	S
Problemstilling	S	P
Valg af datasæt	P	S
Valg af læringsalgoritme	S	P
Optimeringer og forbedringer	P	P
Sikring mod under- og overfitting	S	P
Performance metrics	S	P
Konklusion	P	S

Table 1: Hovedanvsarsfordeling: Hvor P står for primær, og S står for sekundær. Gruppen har deltaget i alle dele ligeligt.

Contents

1 Introduktion

Denne rapport indeholder arbejdet udført af Gruppe 16, i forbindelse med ITMAL slutprojektet. Projektets formål har været at skabe en model til at forudsige musikgenrer for lydfiler. Til projektet har gruppen selv bygget sit datasæt med filer og inspiration fra Kaggle-projektet "GTZAN Dataset - Music Genre Classification"[1]. Rapporten afspejler de overvejelser og opdagelser, der er opnået igennem projektet, og følger gruppens arbejde fra konstruktion af datasæt til performance test af endelige model.

Gruppen har under arbejdet med projektet fået en forståelse af de udfordringer, der findes ved musikgenreklassificering i forhold til dataanalyse, samt hvor meget forskellige genrer minder om hinanden. Dette har haft betydning for det endelige projekt, for selv med respektable resultater har overlapningen mellem genrer vist sig som en udfordring for klassificeringen.

2 Problemstilling

I en verden hvor musik konstant bliver lettere tilgængeligt i form af streamingtjenester som Spotify, Youtube Music m.v. Derfor er det også blevet sværere for mindre populære kunstnere at dele deres musik. Dette har givet en øget musikproduktion over hele verden. I dén forbindelse er det nødvendigt hurtigt at kunne klassificere al den nye musik, der bliver produceret. På den måde er det nemmere for lyttere at finde frem til lige netop den musik de interesserer sig for. Én af disse klassifikationer kunne eksempelvis være musikgenrer.

I dette projekt bliver der undersøgt muligheder for at kunne gøre musikgenreklassificering automatisk, og udfordringerne inden for brugen af den nødvendige teknologi bliver undersøgt.

3 Valg af datasæt

Kvaliteten af den data, man benytter, og mængden af nyttig information, er noget af det mest kritiske man skal have styr på, idet det har en stor betydning for, hvor godt ens læringsalgoritme kan træne. Derfor er det vigtigt at undersøge og lave en grundig forbehandling af ens datadsæt, inden det bliver fodret til en læringsalgoritme.

Det hele kan blive delt ind i tre kategorier:

- Fjernelse ugyldig data og indsætning af værdier der mangler
- Danne en korrekt struktur af data til læringsalgoritmen
- Valg af de rigtige features til modellen

3.1 Håndtering af ugyldig data

Det er ikke unormalt, at der mangler et stykke data i et datasæt. Det kan f.eks. være en samling af spørgeskemaer, hvor det ikke er alle, der svarer på et spørgsmål; værdier der ikke har relevans for det man vil finde ud af m.v. De fleste algoritmer kan ikke håndtere at denne data mangler, og dette kan få den til at give uforudsigelige resultater. Datasættet brugt til klassificering af musikgenrer i dette projekt er 'GTZAN'[1]. Dette datasæt består af flere dataark, der indeholder pre-computed features fra lydfiler, der også indgår i datasamlingen. Der blev taget en beslutning, at gruppen selv skulle udvinde disse features tidligt i projektet, så derfor skulle der laves en analyse af lydfilerne. Her blev der fundet en ugyldig lydfil, hvor det ikke var muligt at udvinde nogen gyldig data.

3.2 Brugen af en korrekt datastruktur

Gruppen har benyttet sig af Pandas DataFrame som den underliggende datastruktur til det, der bliver ekstraheret fra lydfilerne i GTZAN. Et bibliotek ved navnet Librosa er blevet brugt til at få forskellige features fra filerne. Datastrukturen er i sig selv ikke særlig speciel, da det kan ses som et dataark med en kolonne pr. feature. Det eneste, der er blevet ændret fra oprindelige datasæt, er de klassenavne, som er med til at identificere en genre, som er blevet indkodet til numeriske værdier. Dette er gjort i tilfælde af at scikit-learn biblioteket har en intern fejl.

Når datasættet er blevet samlet, bliver det delt ind i trænings- og testdatasæt. Jo større testsættet er, des mere vigtig information bliver der tilbageholdt for læringsalgoritmen. Derfor er det vigtigt ikke at allokere for megen information til testsættet. På den anden side vil et for lille et testsæt øge risikoen for en estimeringsfejl af modellens generelle fejl. Der bliver ofte brugt et forhold på 80:20, 70:30 og 60:40, men dette kommer også an på, hvor stort datasættet, der bliver arbejdet på, er. I dette projekt er der blevet brugt et forhold på 70:30, da der er en del data pr. lydfil.

Featureskalering er også en kritisk del af databehandlingen. Dette er fordi det er de færreste algoritmer, der er ligeglade med skaleringen af features. Der er nogen algoritmer, som er ligeglade med skalering, men det er god skik at gøre det alligevel. På den måde er det muligt at teste flere forskellige algoritmer. Nogle algoritmer, som f.eks. gradient-descent, fungerer også bedre på et skaleret datasæt. I dette projekt er der blevet afprøvet både MinMaxScaler og StandardScaler fra sklearn. Disse metoder er også kendt som normalisering og standardisering.

3.3 Valg af features til modellen

Hvis det bliver observeret, at modellen virker bedre på træningsdatasættet i forhold til test-datasættet, er dette et tegn på overfitting. Overfitting er et begreb der siger, at modellen fitter parameterne for tæt på hinanden i forhold til observeringen i træningsdatassætet. Denne fejl kan blive mildnet ved at bruge regulering og dimensionsreducering. I dette projekt er der blevet testet forskellige metoder. Fra undervisningen kender vi til blandt andet L2/L1-norm, som 'straffer' store individuelle vægte. Fordi der blev dannet et rigtig stort datasæt i starten af projektet, blev der også gjort et forsøg på at reducere dimensionen af datasættet ved brug af KPCA (Kernel Principal Component Analysis). Denne blev valgt fordi korrelationsmatricen (Figur 1) indikerede et ikke lineært forhold mellem de fleste features. Der blev ikke observeret en klar forbedring i de tidlige test-algoritmer, så det var ikke noget, der blev gået videre med. Der blev i stedet fokuseret mere på at reducerer datasættet manuelt ved at fjerne irrelevante features.

Under træning af læringsalgoritmen blev der også testet, om længden af en lydfil havde nogen indflydelse på præcisionen og klassificeringen. Der blev testet på 30/5/3 sekunder. Ud fra dette kunne det konkluderes, at en længere lydfil gav en mere upræcis klassificering over alle genrer. For resultatet af denne test henvises der til bilaget.

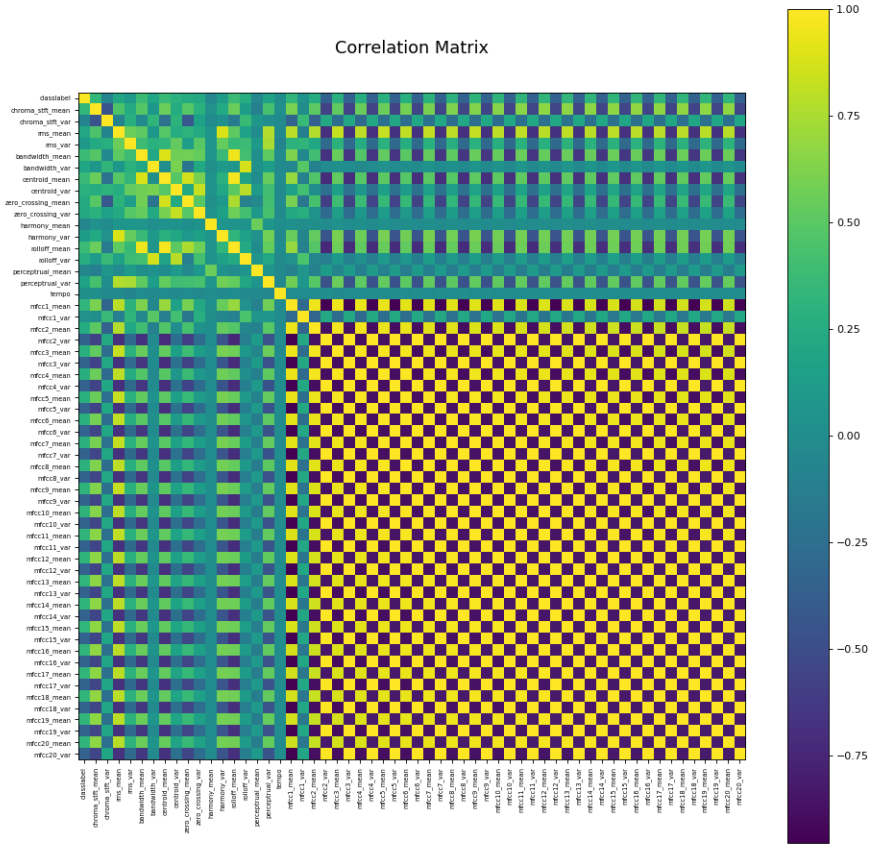

Figure 1: Korrelationsmatrice for det endegyldige datasæt.

4 Valg af læringsalgoritme

Baseret på datasættet blev det hurtigt konkluderet, at dataen ikke var lineær, og selv med nogle kernel tricks til at manipulere dataen blev det ikke bedre. Dette var heller ikke nødvendigvis forventet, eftersom det netop var musik, der skulle arbejdes med, og det endelige mål var genreklassifikation.

Til at vælge den bedste model til datasættet blev accuracy score valgt som det vigtigste for modellen, eftersom der var mange features at tage højde for under forudsigelser, og nøjagtigheden kom derfor til at spille en vigtig rolle. Udover det blev det bestemt, at den valgte model skulle gøre brug af supervised learning, eftersom det gav bedst mening i forhold til det valgte datasæt, hvor der var god mulighed for at danne gode input / output pairs.

Det blev i første omgang besluttet at forsøge sig med et CNN, og til implementering af dette blev der gjort brug af keras. Desværre blev der gentagende gange opnået dårlige resultater, og selv med mindre tweeks til datasættet blev en tilfredsstillende nøjagtighed ikke opnået. Derfor blev CNN droppet, og der blev testet en bred vifte af modeller med deres default hyperparametre, for på denne måde at finde en model, der fungerede godt med datasættet. Resultaterne af disse tests kan ses nedenfor:

```
Accuracy Naive Bayes : 38.539%

Accuracy Stochastic Gradient Descent : 56.156%

Accuracy KNN : 78.345%

Accuracy Decission trees : 66.333%

Accuracy Random Forest : 83.283%

Accuracy Support Vector Machine : 59.393%

Accuracy Logistic Regression : 57.357%

Accuracy MLP : 66.500%

Accuracy Gradient boost : 78.011%

Accuracy Hist gradient boost : 87.654%

Accuracy Cross Gradient Booster : 86.720%
```

Som det kan ses ud fra resultaterne har ensembled decission trees, såsom random forest og gradient boosts, generelt en rigtig høj baseline score på datasættet. Modellen med den bedste score endte op med at være sklearns HistGradientBoostingClassifier[2], men eftersom det stadigvæk er en eksperimentel model i sklearn, blev det valgt at droppe den.

I stedet blev der gjort brug af cross gradient booster (også kaldet XGBoost [6]), som var modellen, der fik den næsthøjeste score. XGBoost er en ensembled model der gør brug af decission trees og en approximate greedy algorithme[4][5]. Eftersom et decission tree ofte kan have svært ved at løse komplekse opgaver alene, giver den ensembled model muligheden for at samle resultater fra mange decission trees og derved skabe sine predictions baseret på de resultater.

Dette resulterer i, at modellens kompleksitet øges i forhold til hvor mange træer der er bundet på modellen. På figur 2 ses den matematiske forskrift for kompleksiteten.

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

Figure 2: Matematisk udtryk for kompleksiteten for XGBoost[5]

På figuren repræsenterer w_j^2 scoren på hvert enkelt blad og T repræsenterer, hvor mange blade (decission trees) XGBoost gør brug af. Dette har en betydning på kompleksiteten af den endelige model, som i dette tilfælde har kunne holdes nede (max dybde sættes til 5, mere om det i optimerings afsnit).

5 Optimeringer og forbedringer

Som nævnt i datasæt afsnittet er der allerede udført forbehandling på dataet så som brugen af MinMaxScaler, som skulle sikre normaliseret data. Udover dette er det også muligt at optimere på læringsalgoritmen, som vil blive beskrevet i følgende afsnit.

Til at gøre dette blev der gjort brug af både GridSearch og RandomizedSearch. Inden for machinelearning findes der to forskellige typer parametre: Dem, der kan læres fra datasættet, som f.eks. vægte i logistiskregression, og så de parametre, der bliver givet til læringsalgoritmen, også kendt som hyperparameter. GridSearch er en brute-force-udtømmende søgningsalgoritme, hvor der bliver givet en liste af forskellige hyperparametre, hvor algoritmen så vil evaluere hver kombination af disse for at finde dét resultatet, der har den bedste ydeevne.

GridSearch er en rigtig værdifuld metode til at finde det optimale sæt af parametre. Dog er evalueringen af alle de forskellige muligheder beregningsmæssigt dyrt. Et alternativ er den nævnte RandomizedSearch, som er givet af scikit-learn. Dette er et paradigme, hvor vi tilfældigt vælger kombinationer af parametre.

Parametrene, der var interessante at afdække, kan ses nedenfor:

```
params = [{
    'eta':[1e-5,1e-3,1e-1,2e-1,3e-1,4e-1,5e-1,1],
    'max_depth':[5,10,20,40,100],
    'subsample':[0.5,1],
    'sampling_method':['uniform'],
    'tree_method':['exact','approx','hist'],
    'grow_policy':['depthwise','lossguide']
}]
```

eta parameteret er også kendt som *learning_rate*, og er vigtigt at få afdækket, da det netop kan hjælpe forebyggelsen af overfitting ved at mindske vægten af nye tilføjet features. I dette tilfælde kunne GridSearch godt have haft endnu flere *eta* værdier, da der er meget store steps mellem hver enkelt værdi.

max_depth parameteret hjælper med at holde styr på dybden af det overordnede træ, og som tidligere nævnt hjælper dette med at holde kompleksiteten af træet nede. Derfor er det også vigtigt at få valgt en god værdi.

subsample parameteret kunne også godt have været afgrænset bedre, men fortæller om, hvor meget af datasættet der skal samples før en iteration af beslutningstræet bygges. Dette hjælper mod overfitting, idet træet kan balanceres før konstruktion.

I dette tilfælde er der kun benyttet ***sampling_method*** uniform, men der findes også en gradientbaseret samplingmetode, som kun fungerer med trætypen gpu_hist, som ikke er blevet brugt, da modellen ikke er opsat til at benytte en GPU til beregninger.

tree_method er hvilken greedy algoritme der skal anvendes. I tilfælde af *exact* bliver alle split kandidater talt, inden konstruktionen af et træ bliver udført. *approx* gør brug af en tilnærmet greedy algoritme og anvender også histogrammer til konstruktionen. Den sidste metode er optimeret version af *approx*, hvor hastighed er vægtet højest.

Sidste parameter der skal afdækkes er ***grow_policy***, som navnet antyder dækker dette over hvordan træet skal vokse. *depthwise* splitter noderne der er tættes på roden af træet når det skal vokse, hvor *lossguide* splitter noderne

som giver det største tab under træning.

Efter at have kørt både GridSearch og RandomizedSearch blev der fundet frem til to vidt forskellige modeller. For GridSearch var den bedste model:

```
best params: {'eta': 0.4, 'grow_policy': 'depthwise', 'max_depth': 5,
'sampling_method': 'uniform', 'subsample': 1, 'tree_method': 'approx'}
best score: 0.8395538395538396
```

og for RandomizedSearch blev modellen:

```
best params: {'tree_method': 'approx', 'subsample': 1, 'sampling_method': 'uniform',
'max_depth': 5, 'grow_policy': 'lossguide', 'eta': 0.5}
best score: 0.8348352236064797
```

Det første der vigtigt at notere her, er at begge modeller har en dårligere score end modellen med default parameter. Dette kan skyldes 2 ting, den første er at GridSearch' *best_score* returnerer middelværdien for kryds validering[3] af den bedste model, derfor er den præsenterede score en bedre repræsentation af den faktiske model. En anden ting er, at det kan skyldes, at datasættet er blevet overfittet. Dette kan nemlig også have en negativ effekt på scoren.

6 Sikring mod under- og overfitting

I sidste afsnit blev det nævnt, at en dårlig GridSearch/RandomizedSearch score kunne skyldes en lettere grad af overfitting. Overfittingen kunne være et produkt af, at parametrene er blevet ændret under GridSearch. Derfor kan læringsalgoritmen bearbejde dataen på en anden måde end med standardparameterne. For at imødekomme det blev læringsalgoritmen refittet, men denne gang med early-stopping, så hvis predictionfejl blev for hyppige, stoppede fittingprocessen. Dette blev gjort ved at benytte valideringsmetoden "merror og mlogloss" som udregner $\frac{wrong-cases}{total-cases}$ og $-logP(y_true|y_pred) = -(y_true * log(y_pred) + (1 - y_true) * log(1 - y_pred)$. Ved at benytte early-stopping på begge modeller, blev der observeret en klar forbedring i accuracy scoren.

Resultatet af GridSearch på læringsalgoritmen efter disse ændringer kan ses nedenfor:

```
Model 1 Accuracy:  86.987%
```

og for RandomizedSearch.

```
Model 2 Accuracy: 86.987%
```

Ved at se på plots fra fittingprocessen kan det konkluderes, om der var tale om overfitting. på figur 3 ses plot for merror validering på model 1.

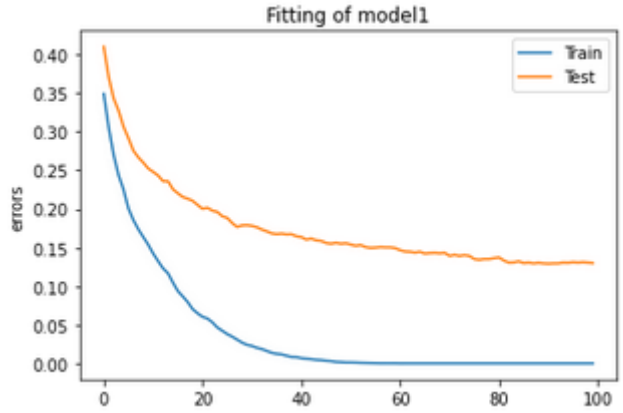

Figure 3: Merror validering plot for gridsearch modellen

og på figur 4 ses plot for logloss.

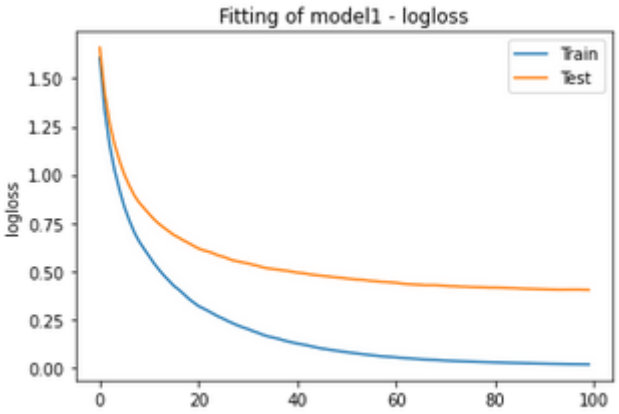

Figure 4: Mlogloss validering plot for gridsearch modellen

Som det kan ses på begge plots er 100% af datasættet benyttet. Derudover kan det ses, at testplottet stadigvæk er aftagende, hvor trænings rammer 0 omkring 40% af datasættet. Derfor tyder det på, at datasættet er underfittet og ikke overfittet som tidligere antaget. Derfor kunne det være muligt at forbedre resultater ved at tilføje mere data. Det blev forsøgt at tilføje mere data til testsættet, men det gav ikke nogen forbedring.

Samme validering blev kørt på RandomizedSearch modellen og på figur 5 ses resultatet for merror validering.

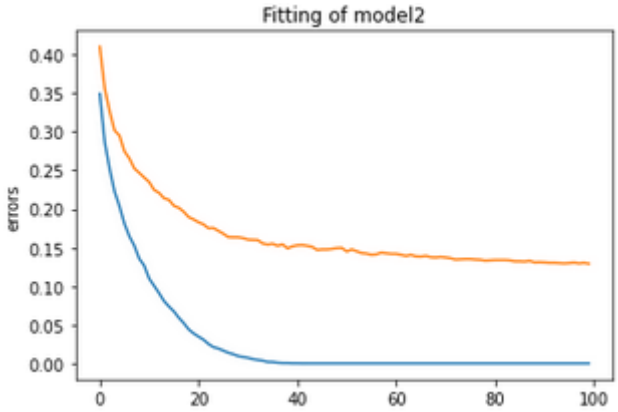

Figure 5: Merror validering for RandomizedSearch modellen

og på figur 6 ses plot for logloss.

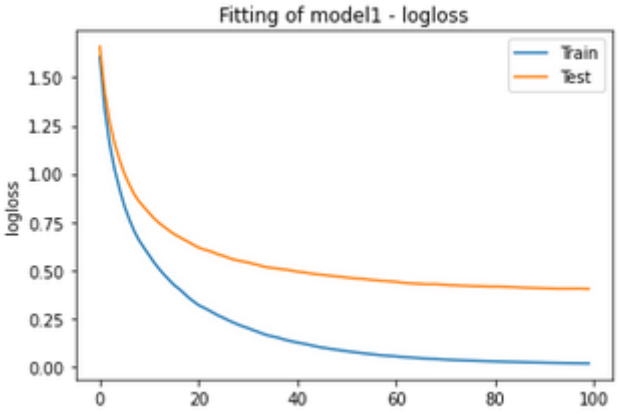

Figure 6: Mlogloss validering plot for RandomizedSearch modellen

Som det kan ses her, er modellen også underfittet, men ved at gøre brug af modellen, kan det ses at træningssættet bliver meget hurtigere fittet.
Der blev også arbejdet med de to tidligere nævnte iterationer af datasættet, hvor filerne var samplet i fuld længde og i stykker af 5 sekunder. I tilfælde af datasættet med fuld længde blev der set et kæmpe dyk i nøjagtighed, og scoren endte på 64.333%, hvilket gav indtrykket af 100 samples pr. genre ikke var nok til at fitte modellen. I tilfældet af 5 sekunder stykker som gav 600 samples pr. genre blev der opnået en nøjagtighed på 82.39%, hvilket lå tættere på det anvendte datasæt.

Confusion matrice samt test af alle modeller kan findes i bilag.

7 Performance metrics

For at tjekke ydeevnen blev der for begge modeller undersøgt for recall og precision samt lavet en forvirringsmatrice. Nedenfor kan resultaterne for GridSearch modellen ses.

`Model 1 Accuracy:86.987%  Recall:86.972%  Precision:86.993%`

og på figur 7 ses confusion matricen for GridSearch modellen.

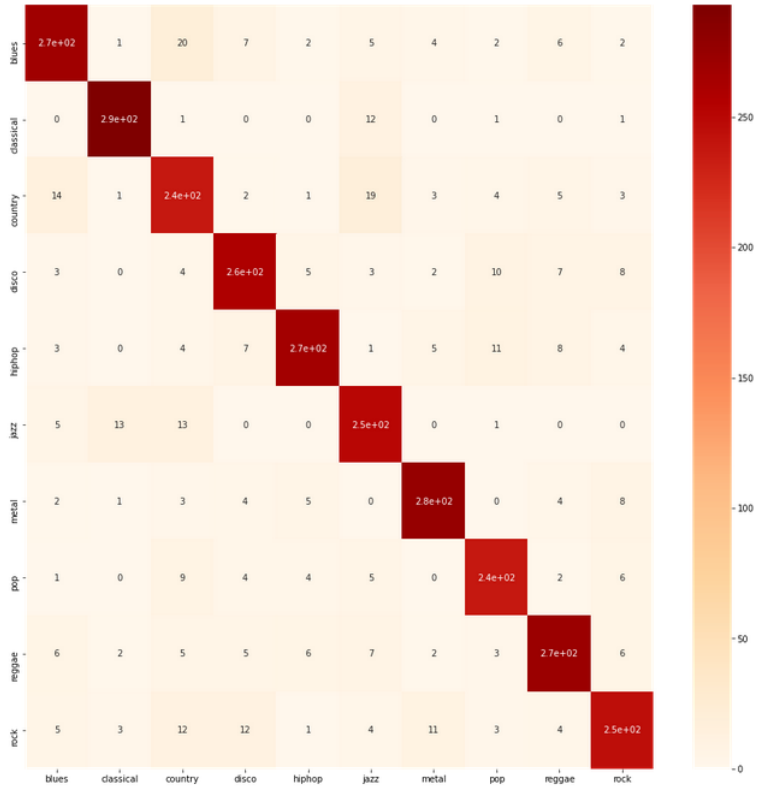

Figure 7: Confusion matrice for model fundet vha. GridSearch

Som det kan ses, har modellen for det meste en del rigtige predictions. De fleste af musikgenrerne ligger i den høje ende af 200, og der er i alt 300 samples i træningssættet for hver genre. Men det er også vigtigt at notere, at der er flere musikgenrer, der har en del fejl i predictions, hvor musikgenrer som pop, country og rock især er hårdt ramt.

Det samme blev gjort for RandomizedSearch modellen, og resultaterne kan ses nedenfor.

```
Model 2 Accuracy:86.987%   Recall:86.984%   Precision:87.037%
```

og på figur 8 ses confusion matricen.

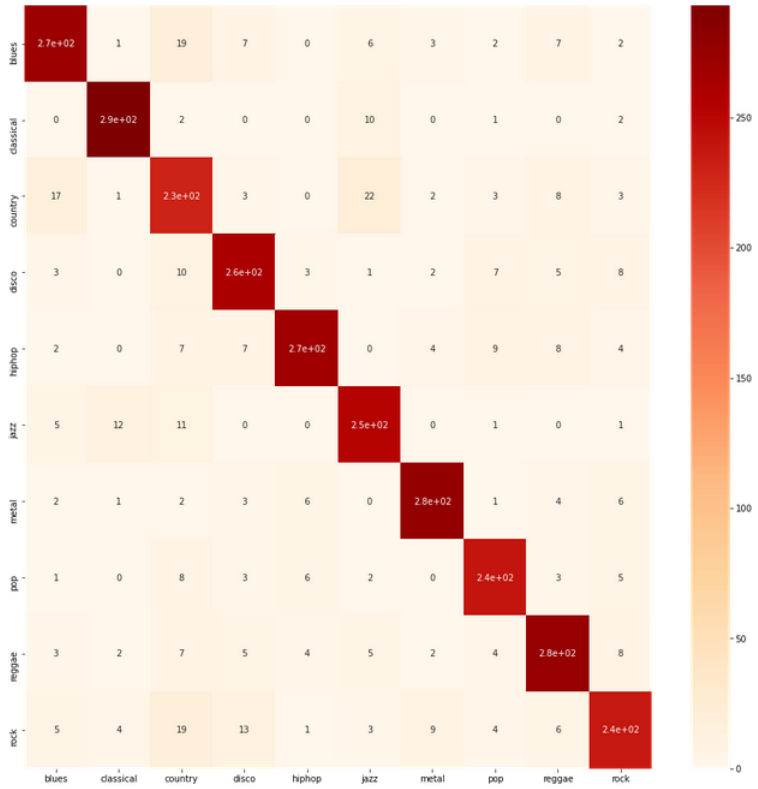

Figure 8: Confusion matrice for model fundet vha. RandomizedSearch

Resultatet på Figur 8 ligner meget resultat for GridSearch, dog er modellen en lille smugle svagere i forhold til country og reggae.

Ud fra matricerne kan det forventes at modellen for det meste vil lave en del rigtige predictions, men andre gange vil den være helt ved siden af.

Dette kan være på grund af den valgte algoritme, som benytter sig af en greedy approximation. Derved kan resultaterne blive påvirket, da den automatisk søger finder de nærmeste lignende værdier, som kan være outliers.

7.1 Test af predictions

Som afsluttende tests af modellen blev der forsøgt med predictions på nogle af musikfilerne fra datasættet. Filerne blev indlæst i sin fulde længde, hvorefter features blev ekstraheret. Resultaterne kan ses nedenfor.

```
Expect:rock, got:['rock']
Expect:pop, got:['pop']
Expect:reggae, got:['pop']
Expect:classical, got:['classical']
Expect:blues, got:['pop']
```

Som det kan ses er nogle af predictions rigtige, men der er også forkerte iblandt. Dette giver god mening i forhold til forvirringsmatricen på Figur 7, hvor det også blev set, at flere af musikgenrerne lå spredt omkring flere genrer

i godt 15% af tilfælde. Derudover er predictions lavet på features for den fulde længde af sangene, men bliver sammenlignet med samples i 3sekunder stykker. Dette kan også have indflydelse på predictions, eftersom sange ikke har konstante værdier igennem hele spilletiden.

7.2 Sammenligning med vedlagt datasæt

Da det anvendte datasæt var konstrueret af gruppen, kunne det være interessant at se, hvor godt det var i forhold til det vedlagte fra Kaggle[1]. Derfor blev GridSearch modellen fittet med datasættet, hvorefter ydeevnen blev testet. Datasættet der blev testet for var 3sekunders stykker ligesom det anvendte.

Resultaterne kan ses nedenfor:

`Model Accuracy:90.257%  Recall:90.280%  Precision:90.224%`

Og på Figur 9 ses forvirringsmatricen for .csv datasættet.

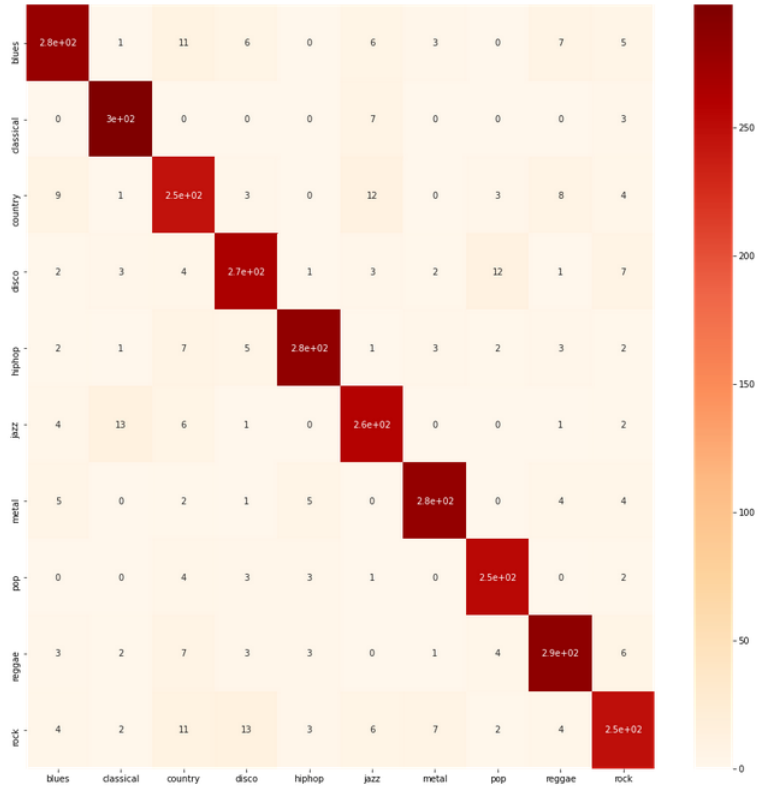

Figure 9: Confusion metrics for model bygget på csv datasæt

Som det kan ses, får modellen en forbedring i nøjagtighed på godt 3% ved at bruge det vedlagte datasæt. Modellen har dog stadigvæk en del fejl predictions, og selvom der ses en lille forbedring, er det langt fra perfekt. Derfor må det konkluderes, at det byggede datasæt har fungeret fint, men det største problem ligger i et begrænset antal samples.

8 Konklusion

Et machinelearningsystem kan ses som en samling af fire forskellige aspekter. Et typisk forløb følger formen af: Forbehandling af data, learning, evaluering og forudsigelse. I dette projekt har vi forbehandlet data. Dette er blevet gjort, idet 'raw'-data sjældent kommer i en form, som er optimal for læringsalgoritmen. Derfor er behandlingen af data et af de vigtigste trin i ethvert form for machinelearningprojekt. I nogen tilfælde er noget data så korreleret, at det er redundant til en hvis grad. Her er det også vigtigt, at man har lavet en god forbehandling af data, som er med til at reducere mængden af plads der bliver brugt på disken. Dette sikrer, at algoritmen kan køre hurtigere. I andre tilfælde kan en reducering af dimensionaliteten af data også give en bedre forudsigelse på uset data. Dette er ofte et tegn på, at der har været for mange irrelevante features eller støj i data. Ydermere for at sikre, at algoritmen også har en god ydeevne, er der blevet brugt tilfældig inddeling af trænings- og testdatasæt. På den måde har det været muligt at fokusere på at optimere modellen, inden vi har skulle teste den.

For at sikre at modellen fungerer korrekt i den virkelige verden, er der blevet brugt en række af valideringsteknikker. Ud over dette kan det ikke forventes, at standardparametre af forskellige læringsalgoritmer er tilstrækkelige for vores specifikke problem . Derfor har vi gjort hyppigt brug af hyperparameteroptimeringsteknikker, der hjælper os med at finjustere ydeevnen af vores model.

Målet for dette projekt var at implementere en læringsalgoritme, der var i stand til at klassificere musikgenrer. Under forløbet fandt vi ud af, at en simpel læringsalgoritme ikke var i stand til at opnå den ydeevne, vi var ude efter. Ved at benytte en samling af eksperter til at lave en forudsigelse, opnåede vi en større nøjagtighed. Derudover kan det konkluderes, at den mest avancerede algoritme er begrænset af dén information, den modtager fra træningsdatasættet. Vi har benyttet databehandling og brugt forskellige metoder inden for hyperparametertuning for at få den bedste model, vi kunne.

Ved at bruge ensemblelæringsalgoritmen Gradient-boosting lykkedes det gruppen at åbne en nøjagtighed på cirka 87 procent. Det kan dog konkluderes, at modellen ikke er særlig god til at forudsige specifikke genrer, som har overlap i features (Se Figur 7). Dette kan være på grund af, at mange genrer har træk, som får dem til at ligne hinanden, eller at datasættet har været langt mindre fittet end forventet. I andre projekter har de opnået gode resultater ved at benytte sig af spektrogrammer og bruge billedekendelelse. I en fremtidig udgave af projektet kunne dette tilføjes i en flerlagsmodel.

References

[1] Andrada Olteanu. *GTZAN Dataset - Music Genre Classification*. URL: https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification. (accessed: 10.12.2020).

[2] Scikit-learn. *HistGrandientBoostClassifier*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.HistGradientBoostingClassifier.html. (accessed: 10.12.2020).

[3] Sklearn. *Gridsearchcv*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. (accessed: 10.12.2020).

[4] wikipedia. *Gradient boosting*. URL: https://en.wikipedia.org/wiki/Gradient_boosting. (accessed: 10.12.2020).

[5] XGBoost. *Introduction to boosted trees*. URL: https://xgboost.readthedocs.io/en/latest/tutorials/model.html. (accessed: 10.12.2020).

[6] XGBoost. *XGBoost*. URL: https://xgboost.readthedocs.io/en/latest/. (accessed: 10.12.2020).