Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek

Računarstvo usluga i analiza podataka

SEMINARSKI RAD

Klasifikacija pas / mačka na temelju zvučnog zapisa

Robert Dumančić

Anto Tufeković

Osijek, 2020.

Sadržaj

[1. Uvod 1](#_Toc32765649)

[2. Opis problema 1](#_Toc32765650)

[2.1. Korišteni podaci 1](#_Toc32765651)

[2.2. Korišteni postupci strojnog učenja 2](#_Toc32765652)

[2.2.1. Locally deep Support Vector Machine (LD-SVM) 2](#_Toc32765653)

[3. Opis programskog rješenja 3](#_Toc32765654)

[3.1. Model strojnog učenja 3](#_Toc32765655)

[3.2. Način korištenja API-ja 5](#_Toc32765656)

[3.3. Klijentska aplikacija 7](#_Toc32765657)

[4. Zaključak 8](#_Toc32765658)

# Uvod

Ovaj rad se bavi rješavanjem problema klasifikacije pasa i mačaka na temelju zvučnog zapisa. Klasifikacija je metoda nadziranog učenja gdje je izlazna veličina kategorija poput živ ili mrtav. Nadzirano učenje podrazumijeva učenje modela na podatcima koji imaju i ulaznu i izlaznu veličinu. Cilj je postići zadovoljavajuću preciznost klasifikacije zvučnih zapisa koji nisu sadržani unutar skupa za treniranje na jednu od dvije kategorije.

Izvršiti će se web aplikacija koja će omogućavati učitavanje vlastitih podataka za koje će prethodno istreniran model u Azure ML Studio vraćati pripadnost jednoj od klasa.

# Opis problema

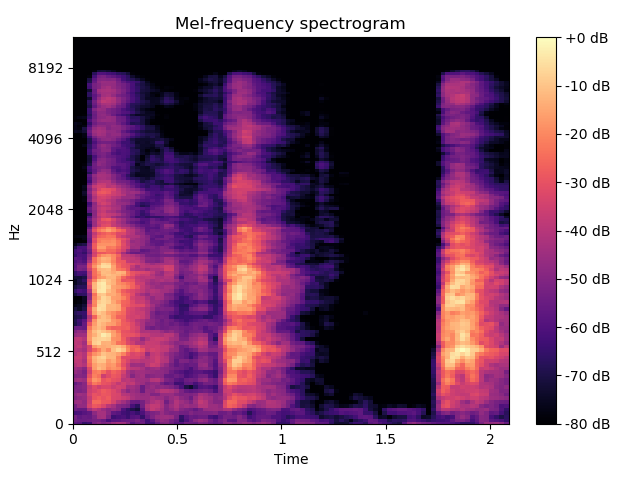
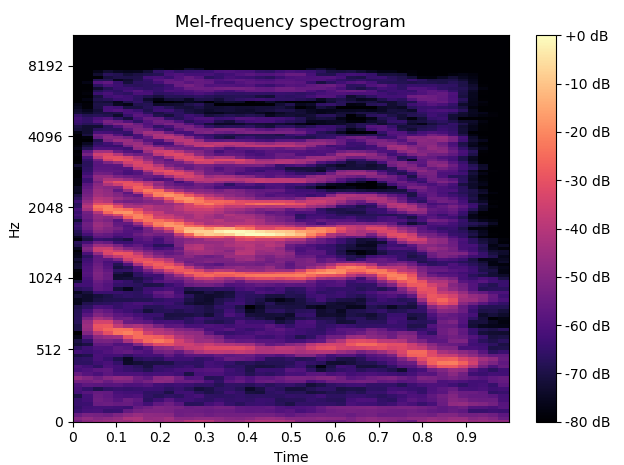
Problem se sastoji od raspoznavanja zvučnih uzoraka na snimci kako bi se mogla dodijeliti pripadnost odgovarajućoj klasi. Moguća uporaba bi bila za pomoć gluhim osobama ili za prepoznavanje na nejasnim zapisima ili zapisima sa šumom.

Rješenja za ovakav problem već ima na stranicama poput GitLab i GitHub, ali nijedan od prethodnih rješenja nema API povezan sa svojim modelom već samo istreniran model u python programskom jeziku. Detaljniji opis problema, u kojim se područjima pojavljuje, kratak prikaz nekoliko sličnih pristupa koji već postoje za isti ili slične probleme.

## Korišteni podaci

Korišteni skup podataka preuzet je s Kaggle (<https://www.kaggle.com/mmoreaux/audio-cats-and-dogs>). Skup podataka sadrži 164 .wav datoteka za mačke i 113 .wav datoteka za pse. Neki od zvučnih podataka su odbačeni jer su sadržavali zvukove obje klase ili su bili potpuno nevezani za klase. Kako bi obrađivali takve podatke moramo izvući značajke prema kojima možemo raspoznavati pse od mačaka. Problem je bio pronaći značajke po kojima se može utvrditi razlika između pasa i mačaka.

Za uvid u značajke u python programskom jeziku korištena je librosa biblioteka. Ona sadrži sve potrebne alate za audio analizu. Jedan od njih je i melspectrogram. Melspectrogram pokazuje intenzitet zvuka uz frekvenciju u vremenu. Jedan je od glavnih alata koje koriste istraživači pri vizualizaciji životinjskih glasanja.



Slika 2.1 melspectrogram za mačku (lijevo) i psa (desno)

Pogledom na sliku 2.1 vidi se kako mačke imaju vrlo dobru harmoničku strukturu koja traje dugo, s najnižim harmonikom na većoj vrijednosti od pseće. Psi imaju najniži i najviži harmonik na nižoj granici od mačje, ali je trajanje iznimno kratko. Tu se vidi znatna razlika između pasa i mačaka po kojima možemo trenirati model. Librosa sadrži funkciju koja vraća vrijednosti MFCC ( Mel Frequency Cestral Coefficients) koji će se koristiti za učenje.

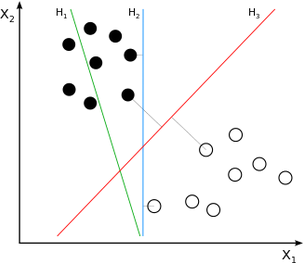
Unutar scripts foldera u projektu se nalazi convert.py datoteka koja sadrži python kod u kojemu su izvučene značajke iz svih podataka korištenog skupa podataka i onda spremljene u .csv datoteku. Podatci su spremljeni kao vektor kojemu prva vrijednost predstavlja klasu pripadnosti, a ostalih 20 su MFCC koeficijenti podataka.

## Korišteni postupci strojnog učenja

Zbog prirode izlučenih značajki se ne zna intuitivno koji se model treba koristiti, te su provedeni eksperimenti nad raznim vrstama modela kako bi se našao najbolji model za učenje. Isprobani su sljedeći modeli: perceptron, šuma odluke, prašuma odluke, logistička regresija, jednostavna neuronska mreža, SVM (support vector machine) i locally deep SVM. Svi su modeli trenirani nad ulaznim skupom podataka koji je nasumično podijeljen na 80% na skup za učenje i 20% na skup za validiranje nad raznim postavkama hiperparametara podešenima pomoću „*Tune model hyperparameters*“ bloka. Korišteni su osnovni parametri samog bloka i svaki model je imao svoj raspon hiperparametara s kojim je blok rukovao te uzimao nasumične vrijednosti unutar raspona hiperparametara zadanih u mdelima. Blok je vrednovao točnost za performanse modela, te se na kraju došlo do zaključka da je najbolji model za klasifikaciju locally deep SVM.

### Locally deep Support Vector Machine (LD-SVM)

LD-SVM je ekstenzija normalnih SVM modela. SVM algoritam pripada nadziranim algoritmima, što znači da se pri treniranju koriste podaci koji imaju ulazne i izlazne veličine pravilno klasificirane. Prilikom treniranja SVM model uči predvidjeti gdje će se nalaziti sljedeći podatak tako što pri treniranju kreira neku višedimenzionalnu plohu odluke. Ta ploha se pri treniranju transformira na takav način da je udaljenost tj. separacija između plohe i skupova podataka što veća. Na taj način kad evaluira novi podatak može ga smjestiti u ovisnosti o plohi odluke.



Slika 2.2

Na slici 2.2 se vidi postupak učenja plohe odluke nad dvodimenzionalnom skupu podataka. Ploha H1 je slabo trenirana (*underfit*), ploha H2 je usred treninga (odvaja podatke ali nije maksimizirana udaljenost međusobno od skupova) dok ploha H3 je na kraju treninga gdje se vidi maksimizirana udaljenost od oba skupa podataka.

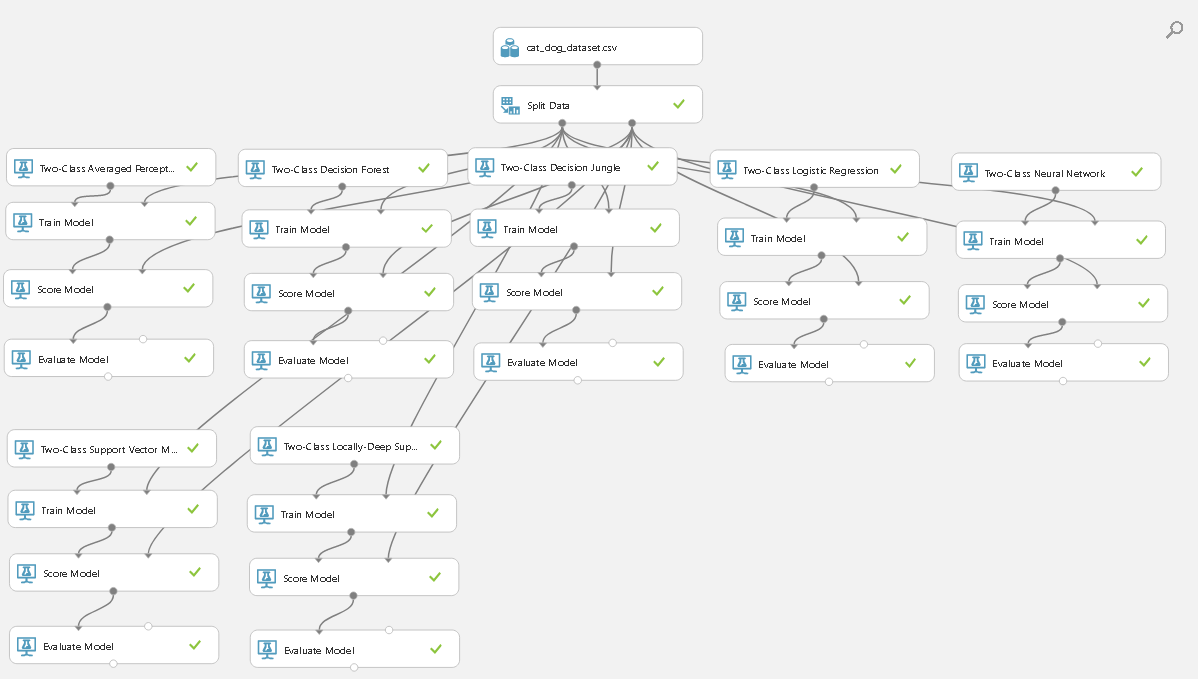
# Opis programskog rješenja

Programsko rješenje se sastoji od više dijelova:

* Azure ML Studio koji se koristi za treniranje i predikciju novih podataka preko web servisa
* Lokalni frontend i backend koristeći python programski jezik
  + Frontend se sastoji od webstranice koju daje python biblioteka Atlas. Atlas se koristio za laganu i jednostavnu uspostavu GUI s python programskim jezikom
  + Backend se sastoji od više funkcija (najviše od python biblioteke za analizu zvukova Librosa) kojom transformiramo podatke u oblik koji Azure ML Studio može obrađivati

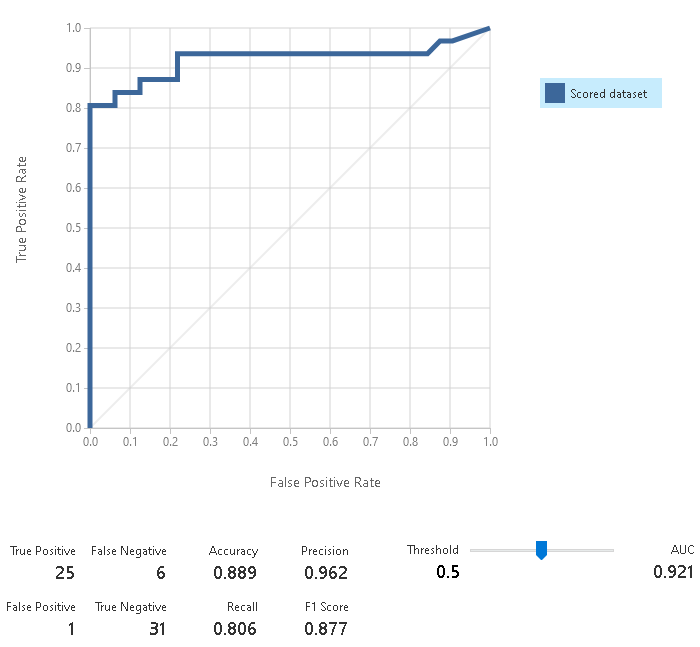
## Model strojnog učenja

Na slici 3.1 su prikazane svi modeli strojnog učenja koji su testirani nad zadanim podatcima. Podatci su dodani u Azure ML Studio kao .csv koji je prethodno generiran i sadrži jedan stupac koji predstavlja *class label* podatka i 20 stupaca koji su njegove značajke. Pomoću bloka *Split Data* podatci su podijeljeni na 80% koji se koriste za treniranje modela i 20% koji se koriste za validaciju. Zbog malog broja podataka u skupu i velikog broja *outliera* čak i nakon filtriranja neupotrebljivih podataka teško je dobiti veliku preciznost modela.

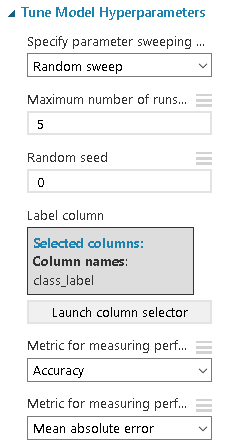
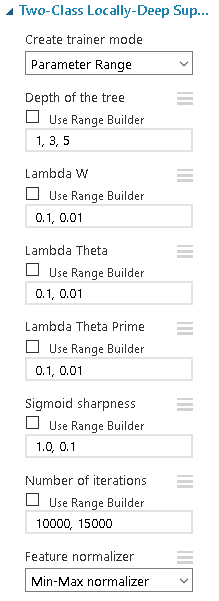


Slika 3.1 Prikaz isprobanih modela

Model korišten za strojno učenje je *Locally Deep Support Vector Machine* (LD-SVM). Dok su ostali modeli davali preciznost između 81 i 85%, LD-SVM je imao najveću preciznost s 89% s optimalnim parametrima. Tijekom korištenja bloka *Tune Model Hyperparameters*, koristile su se osnovne zadane postavke jer je tada bila nepoznata učinkovitost svih pregledanih modela. Tijekom traženja parametara ispada da je optimum bio blizu početnih zadanih vrijednosti samog modela LD-SVM-a. Bilo je potrebno samo u algoritmu promijeniti hiperparametar dubine stabla. Povećanjem ili smanjenjem te vrijednosti od preporučene uzrokuje smanjenjem preciznosti modela.



Slika 3.2 Vizualizacija procjene modela



Slika 3.3 Korištenje *Tune Model Hyperparameters* za pronalazak optimalnih parametara

Prilikom treniranja i prilikom korištenja bloka *Tune Model Hyperparameters* najvažnija metrika bila je točnost, jer je najvaćnije da model dobro predviđa klasu uzorka koji mu je dan.

## Način korištenja API-ja

API koji je pružao Azure Machine Learning Studio se koristi pomoću koda unutar REQUEST/RESPONSE linka koji se nalazi na stranici kada se uspostavi web servis. Korišten je python kod za izvođenje komunikacije sa API-jem.

Prvo bi sa klijentske aplikacije došao URL link koji pokazuje na zvučni zapis (koji može pročitati soundfile biblioteka) koji bi se predao funkciji u isječku koda 3.1.

|  |
| --- |
| def fetch\_file(file\_url):      '''      Give it a file url for a cat or dog .wav file and it will return a string 'cat' or 'dog' depending on the results of the ML prediction      '''      print(file\_url)      if(urlparse(file\_url)[0]!=None):#urlparse to see if it really is a valid url          if(str(file\_url).find("http") == -1):#check to see if it contains the http:// part of the link              temp\_url = "http://"+file\_url          else:              temp\_url = file\_url          try:              data, samplerate = sf.read(io.BytesIO(urllib.request.urlopen(temp\_url).read()))              print("Sample rate:", samplerate)              features = utils.extract\_mfcc\_features\_ds(data, samplerate)              sample = utils.prepare\_sample(features)              #print(sample)              result = azure\_ml\_prediction.make\_request(sample)              if(result.rfind(b'dog') != -1):                  print("dog")                  return "dog"              elif(result.rfind(b'cat') != -1):                  print('cat')                  return "cat"              else:                  print("failed due to bad predict")                  return "failed..."          except:              print("failed due to bad process")              return "unsupported sound or .wav format :(" |

Isječak koda 3.1 Funkcija koja vraća predikciju modela

Iz URL-a bi se povukao zvučni zapis, dobili bi odmah podatke i frekvenciju uzorkovanja. Ti se podaci šalju dalje u librosa funkcije, prikazane u isječku koda 3.2, kako bi se izvršilo izvlačenje značajki (MFCC) iz zvučnog zapisa.

|  |
| --- |
| def extract\_mfcc\_features(file\_name):      '''      Extracts mfcc features from a selected file for machine learning.        Defaults to 20 features.      '''      audio, sample\_rate = librosa.load(file\_name, res\_type='kaiser\_fast')      mfccs = librosa.feature.mfcc(y=audio, sr=sample\_rate)      mfccs\_processed = np.mean(mfccs.T,axis=0)      return mfccs\_processed |

Isječak koda 3.2 korištenje librosa biblioteke za izvlačenje značajki

Nakon što su značajke izdvojene pripremaju se za slanje Azure ML API-ju tako da se slažu u polje gdje je prva vrijednost samo naziv klase i za njom su 20 značajki koje su prethodno dobivene, što se vidi u isječku koda 3.3.

|  |
| --- |
| def prepare\_sample(mfccs\_processed):      '''      Prepares a librosa mfcc field for sending via JSOn      '''      temp = []      temp.append("value")      temp.extend(mfccs\_processed)      return temp |

Isječak koda 3.3 Pripremanje uzorka

Konačno se dobiva rezultat te se sprema u result varijablu. Ta varijabla je odaziv API-ja koji nam je dao predviđanje klase (isječak koda 3.4)

|  |
| --- |
| def make\_request(sample):      '''      Sends a JSON request to a Azure Machine Learning Studio deployed web server for classification predictions.      '''      data =  {            "Inputs": {                    "input1":                  {                      "ColumnNames": ["class\_label", "feature0", "feature1", "feature2", "feature3", "feature4", "feature5", "feature6", "feature7", "feature8", "feature9", "feature10", "feature11", "feature12", "feature13", "feature14", "feature15", "feature16", "feature17", "feature18", "feature19"],                      "Values": [ [ "value", str(sample[1]), str(sample[2]), str(sample[3]), str(sample[4]), str(sample[5]), str(sample[6]), str(sample[7]), str(sample[8]), str(sample[9]), str(sample[10]), str(sample[11]), str(sample[12]), str(sample[13]), str(sample[14]), str(sample[15]), str(sample[16]), str(sample[17]), str(sample[18]), str(sample[19]), str(sample[20]) ], ]                  },        },              "GlobalParameters": {      }          }        body = str.encode(json.dumps(data))        url = 'service\_url'      api\_key = 'api\_key' # Replace this with the API key for the web service      headers = {'Content-Type':'application/json', 'Authorization':('Bearer '+ api\_key)}        req = urllib.request.Request(url, body, headers)        try:          response = urllib.request.urlopen(req)            result = response.read()          print(result)          return result      except urllib.error.HTTPError as error:          print("The request failed with status code: " + str(error.code))            print(error.info())            print(json.loads(error.read()))          return json.loads(error.read()) |

Isječak koda 3.4 request/response

Dobiveni rezultat izgleda kao sljedeće:

b'{"Results":{"output1":{"type":"table","value":{"ColumnNames":["class\_label","feature0","feature1","feature2","feature3","feature4","feature5","feature6","feature7","feature8","feature9","feature10","feature11","feature12","feature13","feature14","feature15","feature16","feature17","feature18","feature19","Scored Labels","Scored Probabilities"],"ColumnTypes":["String","Double","Double","Double","Double","Double","Double","Double","Double","Double","Double","Double","Double","Double","Double","Double","Double","Double","Double","Double","Double","String","Double"],"Values":[["value","-306.11078","63.634872","-18.743687","-12.35584","-6.8618484","-7.234913","-6.3914943","-7.957151","-6.3498583","-5.3261323","-1.1621065","0.8609682","6.066177","9.224042","6.138829","0.39999673","-0.9925317","1.6467074","0.32815915","-1.7941998","cat","0.0448621511459351"]]}}}}'

U ovom prikazu se vidi da je u zadnjem popisu podataka dodana labela koja nam govori predviđanje API-ja te moramo samo provjeriti koja je klasa (dog ili cat).

## Klijentska aplikacija

Klijentska aplikacija se sastoji od web-stranice koja je omogućena putem python biblioteke Atlas kao što je prikazano na slici 3.4.



Slika 3.4 Prikaz web stranice

Biblioteka Atlas se koristi za jednostavno dodavanje grafičkog sučelja putem HTML stranice koja se besplatno pruža preko Atlas Toolkit-a (sve je ugrađeno, samo se pokrene funkcija *Atlas.launch(callbacks, None, head)* kako bi se pokrenula web stranica koja se automatski host-a na internetu). Biblioteka radi na način da prima HTML stranicu koja se umeće u Atlas-ovu stranicu, nad kojom se onda mogu pozivati funkcije (npr. dugme “Submit” ima funkciju koja uzme URL koji se nalazi u “Input” polju za unos, te proslijedi ju funkciji koja je navedena (*fetch\_file()*) koja je navedena u isječku koda 3.5 te očisti polje za unos.

|  |
| --- |
| def acSubmit(dom):      dom.focus( "input")      print(dom.getContent("input"))      dom.setContent("result", "Result: ...apraising..." )      dom.setContent("result", "Result: "+fetch\_file(dom.getContent("input")))      dom.setContent("input", "") |

Isječak koda 3.5

# Zaključak

U ovom radu analiziran je skup podataka zvučnih zapisa koji sadrži razna glasanja mačaka i pasa. Na osnovi tih podataka pronađene su značajke koje razdvajaju dvije klase i istreniran je model koji će onda moći vršiti predikciju na nepoznatim podacima. Zbog malog skupa podataka kao i velikog broja suvislih i neupotrebljivih podataka model nije mogao dostići visoku preciznost bez overfitting-a. Od više ispitanih modela izabran je LD-SVM jer je dobio najveću preciznost pri ispitivanju s optimalnim parametrima.

Koristeći Azure ML Studio napravljen je API koji prima nove podatke i vrši predikciju na temelju prethodno istreniranog modela. Primjeri na kojima je testiran rad modela su .wav datoteke s web stranice <http://www.wavsource.com/animals/animals.htm> gdje za 7 primjera psećih zvukova model točno pogodi na 4 (otprilike 60% preciznost).

Proširenjem skupa podataka i uvođenjem novih, raznolikijih podataka za treniranje bi se moglo znatno proširiti postojeći model i povećati preciznost predikcije modela.

Programskom je rješenju moguće pristupiti preko: [Programsko rješenje na GitHubu](https://github.com/ATufekovic/RUAP_projekt_dumancic_tufekovic)