ERHVERVS AKADEMI AARHUS

ML Eksamensopgave efteråret 2020

Lav så mange opgaver af de følgende som du kan. Jo flere opgaver du kan besvare godt jo bedre.
Der vil blive givet point for kvaliteten af forklaringerne af dine valg undervejs og antallet af opgaver du får besvaret.

Du skal aflevere en individuel rapport enten i .pdf format eller som python notebook (hvis man aflevere en notebook, så skal der vistnok uploades en "tom" .pdf til wiseflow - der kan så evt. bare stå i den at opgaven kan findes i en notebook eller flere notebooks i en .zip fil, som man så uploader til wiseflow. Det er fordi wiseflow ikke kan "genkende" notebook formatet. En .zip fil uploades som det der hedder "Ekstra materiale" i wiseflow)

Derudover skal alle .py filer afleveres i en samlet .zip, hvor eventuelle notebooks også er med i denne samlede .zip fil.

Opgave 1 og opgave 2 kan laves i vilkårlig rækkefølge - de afhænger ikke af hinanden og arbejder også på forskellige datasæt
(dog ikke under-opgaverne, som oftest vil være bedst at lave nogenlunde i den beskrevne rækkefølge)

Du skal aflevere på wiseflow senest 17. december kl 23.59.
Men upload opgaven i god tid.

Opgave 1

I klassen har du arbejdet med en begrænset version af rigtige data fra Titanic.

I denne opgave vil du arbejde med et større datasæt fra Titanic (dog stadigvæk ikke det fulde datasæt) og bruge machine learning på dette datasæt.

Vores label i dette datasæt er "Survived" - altså om en person er overlevet eller ikke. Du skal i denne opgave bruge machine learning til forudsige om en person overlever eller ej, baseret på forskellige oplysninger.

Datasættet som SKAL bruges kan findes på canvas - "titanic_800.csv"

Det består af en header og 800 passagerer. Du kan importere det ind i et program som kan læse .csv filer for at få et overblik (f.eks. openoffice er glimrende til at læse .csv filer).

Her er en forklaring af hvad de forskellige felter i headeren betyder (name og passendId burde være åbenlyse, så er ikke med her):

Variable i data	**Definition**	**Nøgle**
Survival	Survived or not	1 = survived, 0 = died
pclass	Passenger class	1 = 1st class, 2 = second class, 3 = third class
Sex	Sex	Male or female
age	Age in years	
sibsp	Number of siblings and spouses aboard the Titanic	

parch	Number of parents and children aboard the Titanic	
Ticket	A ticket number	
fare	The price paid for the ticket	
cabin	The cabin number	
embarked	Port city where the passenger embarked	C = Cherbourg, Q = Queenstown, S = Southampton

I det følgende er der forskellige sektioner (markeret med fed skrift). I hver sektion kan der være flere spørgsmål. Din opgave er at besvare så mange spørgsmål så godt du kan.

Du skal også aflevere din(e) .py fil(er) for denne opgave eller notebooks med din kode.

A) Forstå problemet og data

Hvilken slags ML problem taler vi om med dette datasæt? Vær så præcis som muligt.

Hvor mange features indeholder dette datasæt? (vi tæller ikke en label som en feature)

Hvor mange samples indeholder dette datasæt?

B) Visualisering af data og statistik

Undersøg datasættet ved brug af simpel statistik. Det vil være en fordel at bruge panda dataframes til at håndtere data.

Visualiser de ting du mener er relevante for at forstå data bedre - f.eks. som boxplot eller som scatterplot, hvis du vil vise en sammenhæng mellem to variabler.

Er der nogle stærke (lineære) korrelationer mellem noget af data?

C) Rensning og klargøring af data

Vi har en label “Survived”, som er det rigtige resultat vi kan bruge til at træne efter, dvs. en label.. Men i forhold til de andre features, så skal du tænke over hvilke features du vil beholde og om der er nogle features, som du kan smide væk?

Det kan selvfølgelig være “farligt” at smide features væk, men i dette datasæt er der feature(s) som “roligt” kan smides væk. Argumenter for dine valg.

De/den feature(s) du fjerner vil så ikke være en del af trænings og testsættet.

Nogle af features har manglende data - f.eks. age har nogle manglende data. Hvis du vælger at beholde en feature og den har manglende data er du nødt til at have en måde at håndtere disse data på. Du skal forklare hvordan du gør.

Du er også nødt til at håndtere at nogle data ikke er numeriske data (dvs. ikke tal) og hvis du skal bruge dem skal de oversættes til tal. Der er forskellige måder at gøre det på (se bogen), men ved simple binære værdier som f.eks “Sex” som kan være female eller male kan man roligt oversætte dem til 0 og 1 (i hele datasættet)

Det kan gøre på mange måder - f.eks. med pandas findes der en replace :: `data['Sex'] = data['Sex'].replace(['female'],1.0 )`

Dette vil ændre female til 1.0 (male skal så også ændres til 0.0)

En anden (smartere) måde er at bruge label encodereren:

(https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html)

```
lb = LabelEncoder()

data["sex"] = lb.fit_transform(data["sex"])
```

Her skal man være opmærksom på at hvis man har f.eks. 3 labels, får de værdierne 0, 1 og 2, og så skal man huske at normalisere disse værdier også (hvis man bruger normalisering)

Efter cleaning (og måske skalering/normalisering af data? Vil du bruge dette?) - hvor mange samples og hvor mange features har du så tilbage? (og hvilke features)

Du skal nu splitte dit datasæt op i to dele - et træningssæt og et testsæt.

Dit træningssæt vil bestå af xtrain (features) og ytrain (labels) og dit testsæt vil bestå af xtest (features) og ytest (labels).

Som udgangspunkt er der 800 samples i dette datasæt (måske/måske ikke har du mindre efter cleaning).

Hvor mange samples vil du putte i træningssættet og hvor mange i testsættet og hvorfor?

Du bestemmer selv om du vil splitte data tilfældigt (men så brug randomstate til en fast værdi for at kunne sammenligne fra gang til gang om dine resultater bliver bedre) eller om du vil beholde den sidste "del" af datasættet som testdata.

d) Vælg en ML model og træn på data.

Vælg mindst en ML model og træn din model på dit træningsdata. Forklar, hvorfor du har valgt denne ML model?

e) Evaluating performance on the test set.

Evaluer din models performance på testsættet.

Angiv precision og recall og F1 score for din model.

Forklare også hvad er forskellen på precision og recall med dine egne ord - altså hvad symboliserer precision og recall egentlig? Hvad betyder F1 scoren?

Angiv også din confusion matrix i din rapport/notebook og brug værdierne fra den til at finde ud af hvad din model er god til og ikke god til? (Hint: Se på de forskellige værdier for TN, TP, FN, og FP, som alle kan findes i din confusion matrix)

f) Experimenter - forbedringer af performance

Prøv at lave mindst et systematisk eksperiment for at se om du kan opnå en højere F1 score på dine test data. (du må gerne lave mere end 1 eksperiment)

Det kan f.eks. være ved at ændre parametrene til den valgte model eller en anden måde at håndtere tomme data på.

Det kan også være ved at prøve en anden ML model og sammenligne F1 score med den model du valgte tidligere.

Dokumenter dine eksperimenter og resultater.

Hvor meget blev F1 score forbedret med din bedste model i forhold til din første model med de første valgte parametre? (eller var der ingen forbedring?)

Virker det “nye” resultat overraskende, eller havde du forventet det?

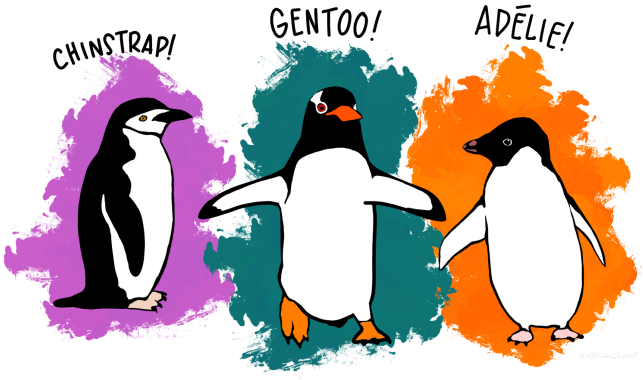
Opgave 2

På canvas ligger et andet datasæt i .csv format.
Filen hedder penguin_size.csv og indeholde data omkring 3 forskellige arter af pingviner.

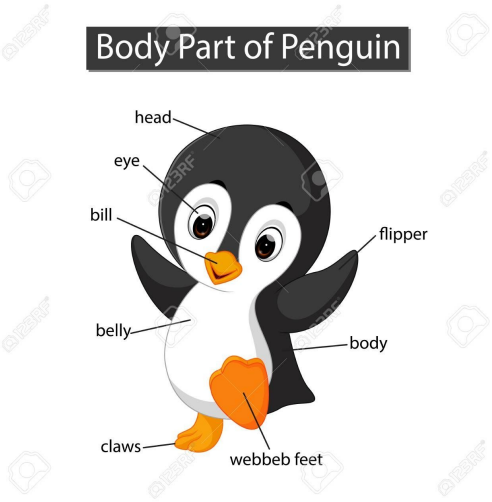
Data indeholder følgende informationer:

- **species**: penguin species (Chinstrap, Adélie, or Gentoo)
- **culmen_length_mm**: culmen length (mm)
- **culmen_depth_mm**: culmen depth (mm)
- **flipper_length_mm**: flipper length (mm)
- **body_mass_g**: body mass (g)
- **island**: island name (Dream, Torgersen, or Biscoe) in the Palmer Archipelago (Antarctica)
- **sex**: penguin sex

Så der er 3 forskellige pingvin arter:



Culmen : Det er overnæbbet, så der er information om længden og bredden af det.
Flipper: Det er en pingvins “arme/vinger” - se skitse nedenunder
Sex: Er Female eller Male eller NA (Non Available - dvs ukendt)



I de følgende sektioner er der nogle delopgaver.

A) Forstå data

Species (arten) er i denne sammenhæng vores label (dvs. vores klassifikation).
Hvor features er det i dette datasæt (label regnes ikke med). ?
Hvor mange samples er der i dette datasæt?

B) visualisering og klargøring af data

Læs datasættet ind i en panda dataframe.

I datasættet er der nogle felter hvor der står NA (i forskellige kolonner), som betyder at der ikke er noget data.

Du er nødt til at vælge en måde at håndtere disse data på - der er forskellige strategier vi har set på. Så vælg en måde at håndtere det på og beskriv hvad du har valgt og hvorfor?

Derefter skal vi have adskilt vores labels fra resten af data.
Det kan f.eks. gøres sådan her (her hedder min dataframe data):

```
labels = data["species"].copy()
data.drop("species",axis =1, inplace=True)
```

Så har vi vores labels i et label array og resten af dataframen indeholder så kun features og ingen labels (vi har jo fjernet dem fra dataframen).

Der er også nogle data som ikke er nummeriske - f.eks. Island, Sex og Species. De skal oversættes til tal - her kan du f.eks. bruge labelencoder igen (se Titanic opgave):

```
lb = LabelEncoder()
data["island"] = lb.fit_transform(data["island"])
```

Vil oversætte alle værdier i island kolonne til 0,1, 2 etc...

Nu kan det være lidt uoverskueligt at arbejde med mange dimensioner.

Så nu skal du bruge SelectKbest på data til at udvælge de 2 bedste features, som er bedst til at forudsige species. Hvor "gode" er disse features - angiv scoren ifølge SelectKbest.

Kunne vi egentlig have brugt PCA i stedet for SelectKbest og hvad er forskellen på disse to ting i dine ord?

Hvilke to features (angive navnene på de to features) er de bedste ifølge SelectKbest?

Derefter skal du visualisere de to bedste features i et scatterplot (dvs. den ene feature bliver x-værdien, den anden feature bliver y-værdien og farven er så den species som den pågældende pingvin tilhører).
Dette kan gøres med (du skal have imports med for f.eks. ListedColormap):

```
colormap = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])

data.plot(legend=False, kind = "scatter", x =
"navn_på_bedste_feature_1", c = labels, y =
"navn_på_bedste_feature_2",
          cmap = colormap)
```

Navnene er søjle navnet i pandaen (her hedder pandaen stadigvæk data) på de to bedste features.

C) Brug af clustering kmeans

Nu har vi godt nok en label her i dette datasæt, kaldet species. Så vi kunne bruge supervised learning. Men vi vil i stedet bruge unsupervised learning på dette datasæt.

Så vi vil prøve at bruge clustering på data fra de to bedste features og så se hvor gode clustering algoritmer er i forhold til de rigtige labels. Dvs. vi kan tjekke hvor god vores clustering er, da vi i dette tilfælde kender det rigtige resultat. Når vores data er i 2D er det også nemmere at foretage en visuel inspektion af vores clustering - derfor reducerer vi også til 2D.

Udfra din viden om data, hvor mange clusters skal vi så forvente at finde, hvis en clustering skal være god?

Paser dette tal nogenlunde med visualiseringen (scatterplot) fra delopgave b) ?

Nu skal du igang med at bruge kmeans på de to bedste features.
Først skal de hives ud fra vores dataframe. Dette kan gøres sådan her:

```
X_best_features = data[['navn_best_feature1',
'navn_best_feature2']]
```

Her referere de igen til søjlen navnene på de to bedste features, som du fik tidligere med selectKBest

Så kan du bruge kmeans:

```
kmeans = KMeans(n_clusters=k,
random_state=42).fit(X_best_features)

labels_kmeans = kmeans.labels_  # for at få labels
```

Så prøv at plotte punkterne igen, men med de labels som kmeans har fundet i stedet for labels angivet i data. Prøv også at plotte cluster centers ind på samme plot.

Sammenlign visuelt de to plots - hvor god synes du kmeans er?

Hvis vi nu gerne ville have et tal for hvor god vores clustering er, har du så nogle ideer til hvad man kunne gøre? (Du behøver ikke at implementere det, men kom med en ide og husk vi kender jo det rigtige resultat i dette tilfælde)

d) Brug af DBScan.

Kmeans virker måske okay, men nok ikke perfekt - (præcist hvor god den er har du fået visualiseret i opgave c).

Tror du at DBSCan vil kunne klare sig bedre på de to bedste features end kmeans? Og hvorfor / hvorfor ikke?

(du behøver ikke implementere DBscan i kode, men giv nogle gode argumenter for dit svar)