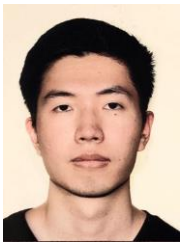
62533 Big Data

Final Report

Chenxi cai
s205420



Alexander solomon
s201172

20. Maj 2022

Indhold

Visualisering

Visualisering er at omsætte en datamængde til noget visuelt og grafisk, hvor læsere kan få hurtigere og bedre mulighed til at fange specifikke information i data. Dette kan hjælpes på vej af en række diagrammer og plots, såsom histogram, piechart, scatterplot og boxplot. Med de visualiseringer fremhæves outlier og tydeliggøres de vigtigste information, dermed skabes et hurtigt overblik over et datasæt.

Beskrivelse af datasættet

Til vores visualiseringsafsnit har vi valgt et datasæt, der er baseret på politirapporter fra San Francisco. Det indeholder forskellige typer af kriminalitet, som er begået i byen, hvor de er begået og hvornår.

Variable informationer:

- Pdid
- IncidntNum
- Incident Code
- Category
- Descript
- DayOfWeek
- Date
- Time
- PdDisctrict
- Resolution
- Address
- X
- Y
- Location

Fremgangsmåde

Vi har valgt at bruge Jupyter Notebook til at lave visualisering af vores datasæt, som er baseret på programmeringssproget Python.
Først og fremmest er vi kommet frem til, at der skal bruges nogle Python-biblioteker, såsom: panda og matplotlib.pyplot. Vi er derfor startet ud med at importerer dataen til disse.
I næste afsnit af rapporten vil vi benytte panda til at manipulere vores datasæt, til bl.a. udrensning af unødvendigt variabler og til at danne nye tabeller.
Dermed vil vi benytte den essentielt Python bibliotek, Matplot til at skabe visualisering af vores datasæt i analyseafsnittet.

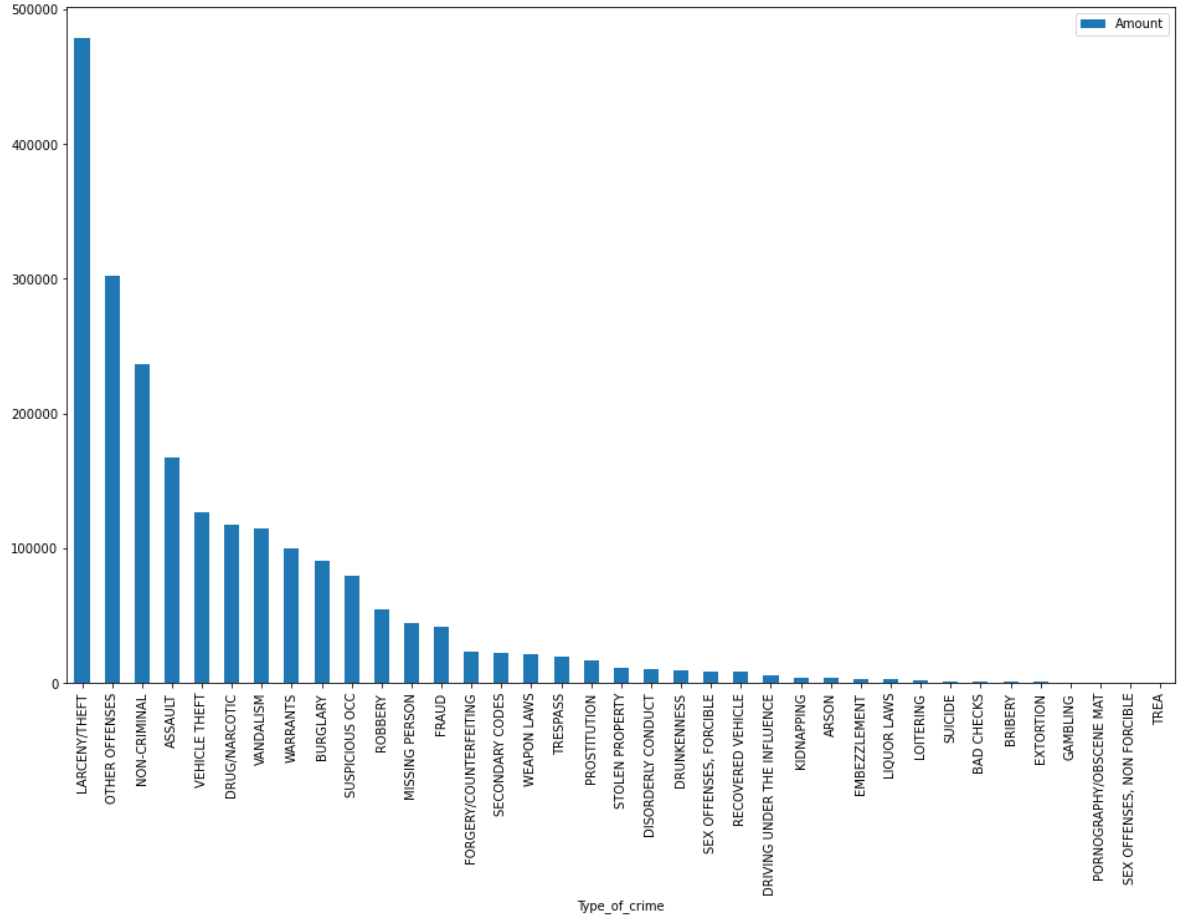
Analyse

Baseret på det udvalgte datasæt, mente vi, at det var mest interessant at undersøge hvilken type kriminalitet, der blev begået mest af og om der nogle dage på ugen, forekommer mere kriminalitet.

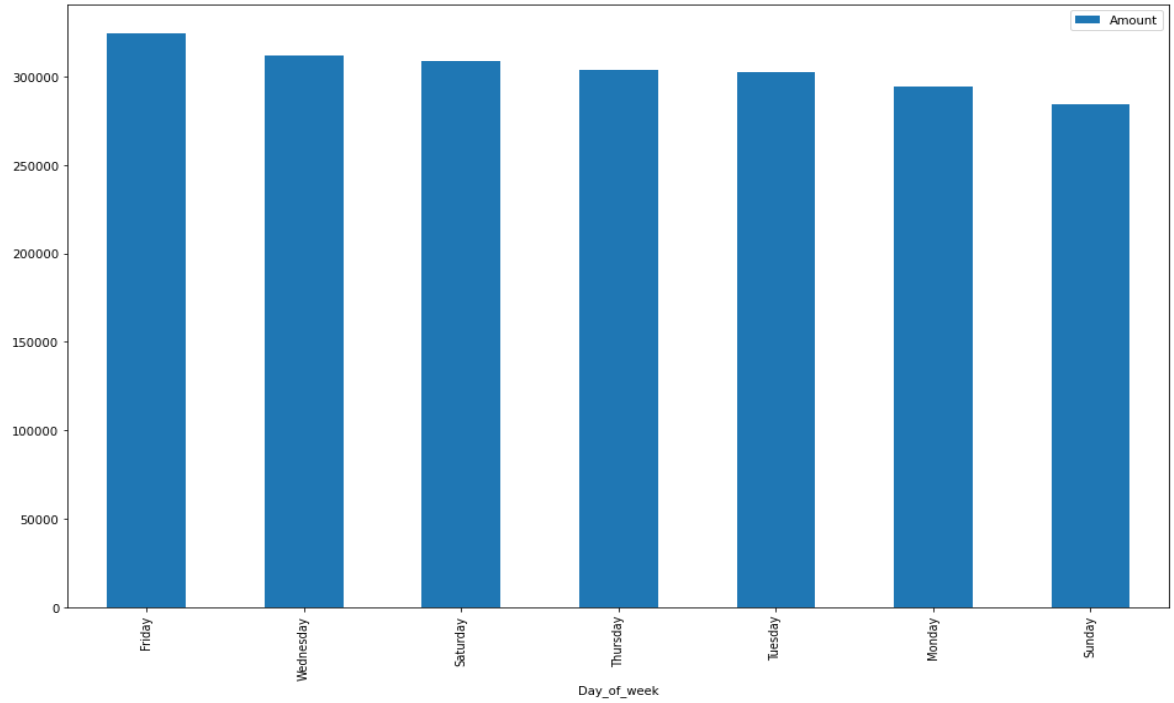
Til at undersøge vores spørgsmål og vise vores resultater, vil vi lave visualiseringer af de data.
Baserer på de udvalgte variabler, har vi dannet et histogram, boxplot, piechart og scatterplot, som hurtigt giver læseren et overblik over de vigtigste informationer. Vi har valgt at basere visualisering på følgende variabler og kategorier: Dayofweek, antal og type af kriminalitet.

Histogram

Nedenfor ses to histogrammer. Det første illustrerer de 37 type kriminalitet (x-aksen) samt hvor ofte de forekommer (y-aksen). Her ses det tydeligt at tyveri er den hyppigst begåede form for kriminalitet i San Francisco, dernæst er det "lovovertrædelser" og "Ikke-kriminel", hvor tyveri er dobbelt så hyppig som "ikke-kriminel".

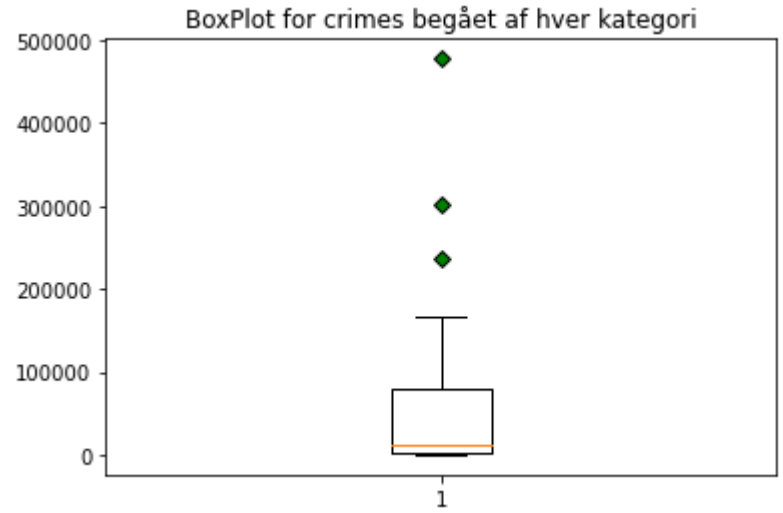


Nedenfor ses et histogram over kriminalitet fordelt på ugens dage. Det ses at der bliver begået mest kriminalitet om fredagen og mindst om søndagen. Forskellen på de to er dog ikke stor, og chancen for at støde på/opleve kriminalitet på en hvilken som helst dag på ugen, er derfor nogenlunde den samme.

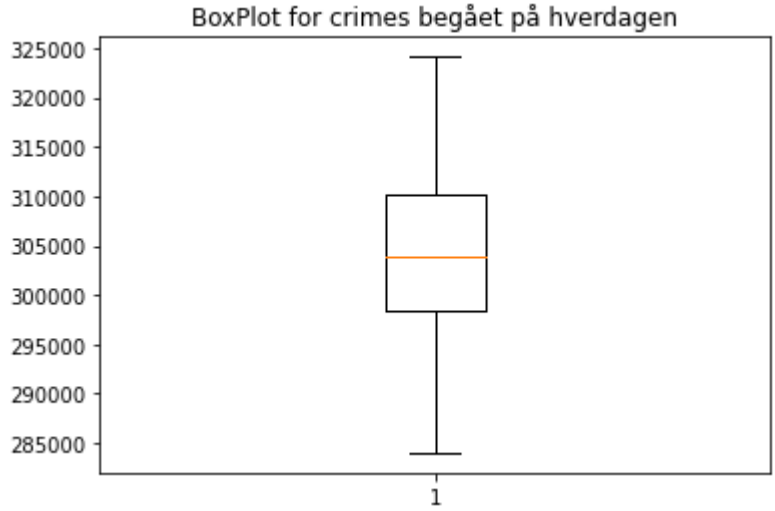


BoxPlot

Nedenfor ses to Boxplot. Her har vi brugt samme data som i histogrammerne, men da de præsenteres på en anden måde, har vi mulighed for umiddelbart at analyserer dem anderledes.
På Boxplottet ses der tre outliers, som er hhv. tyveri, lovovertrædelse og ikke-kriminel. Det ses også at medianen er lav, tættere mod 1. kvartil, hvilket betyder dataen ligger mere kompakt under medianen og mere spredt over.



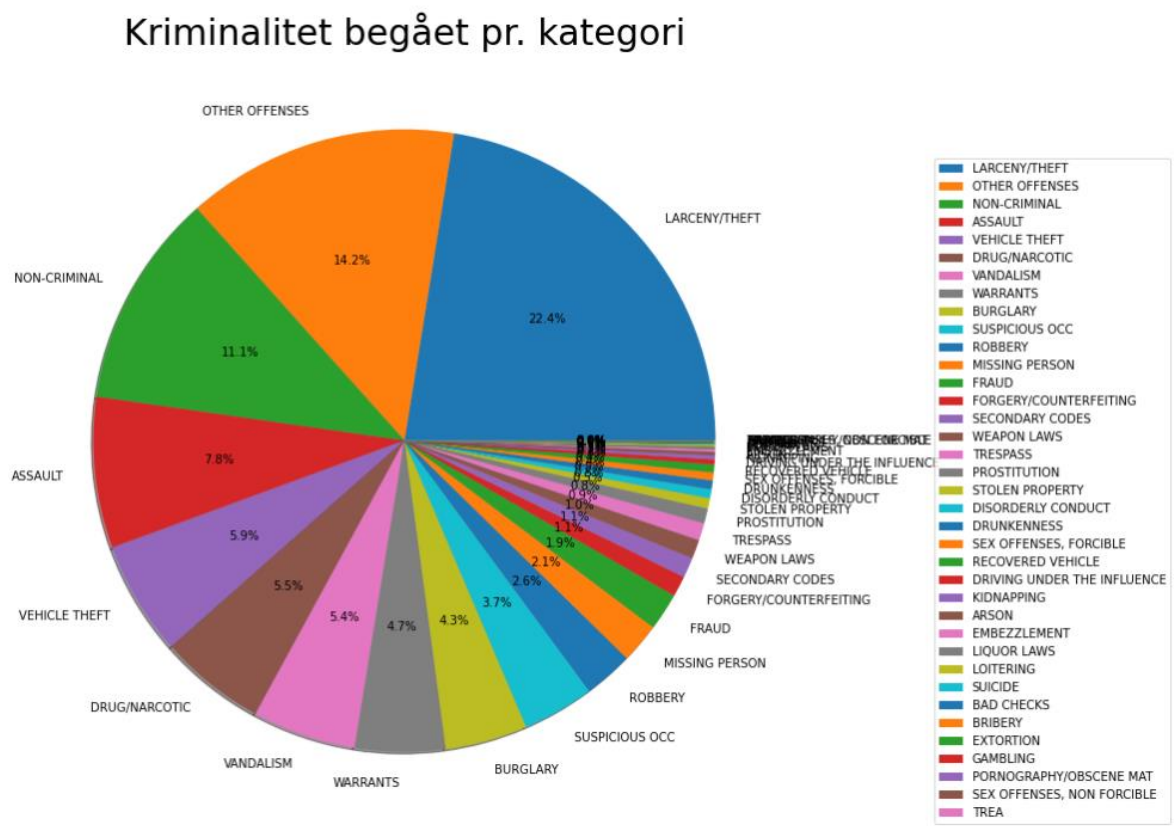
I dette boxplot ses der ingen outlier og medianen ligger i midten. Det hentyder til at dataen er ligeligt fordelt.



Piechart

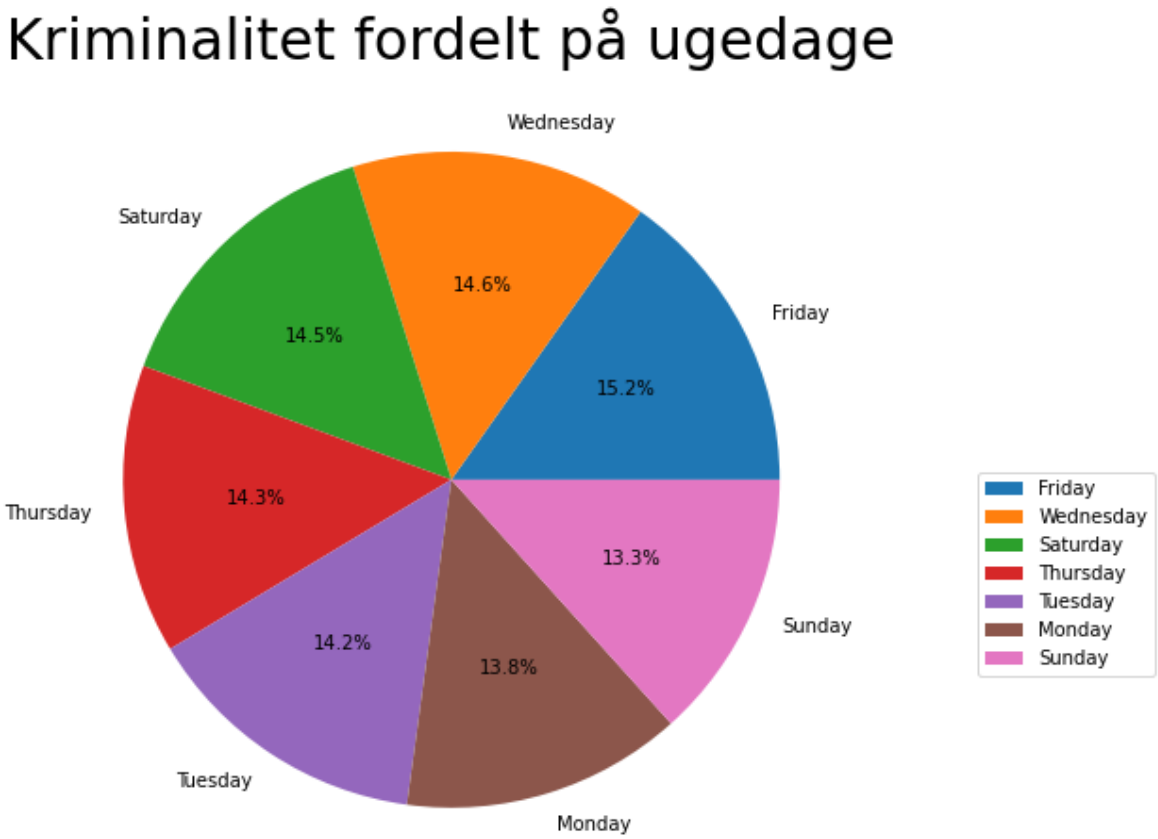
Kriminalitet begået pr. kategori i %

For at give en visualisering i %, har vi lavet et piechart. Pichartet “kriminalitet begået pr. kategori” viser tydeligt hvilken typer kriminalitet, der bliver begået mest af. Det ses at de tre første kategorier står for lidt over 30% af den samlede kriminalitet. Piechartet bliver utydeligt når procentdelen kommer under 2%. Her refereres i stedet til legend kassen, ude i højre side, hvor typerne er placeret i rækkefølge (fra højest til lavest procentfordeling). Her kan de sidste kategoriers procentsats dermed læses.



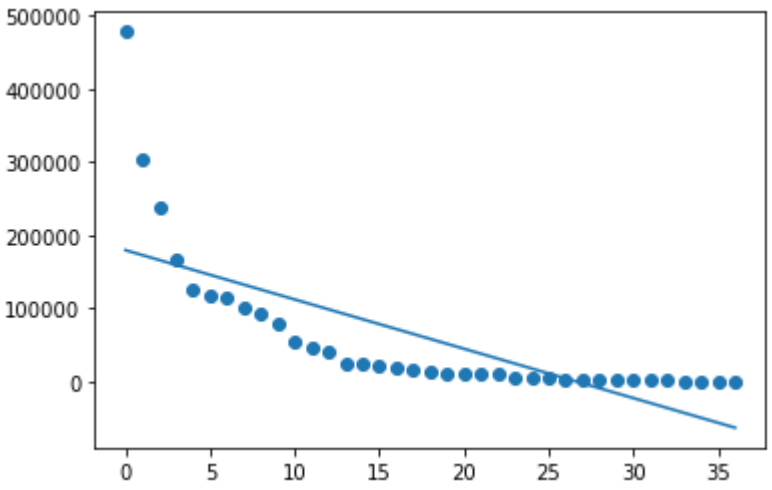
Kriminalitet fordelt på ugedage i %

Hvis man skulle overveje og blive hjemme en dag om ugen, for at undgå diverse kriminaliteter, så skulle det, udfra nedenstående piechart, være om fredagen. Her forekommer hele 15,2% af den samlede kriminalitet, hvilket er den største del. Alle ugens dage er dog indenfor 1.5% af hinanden. Altså er dataen forholdsvis ligelig fordelt, på alle ugens dage.

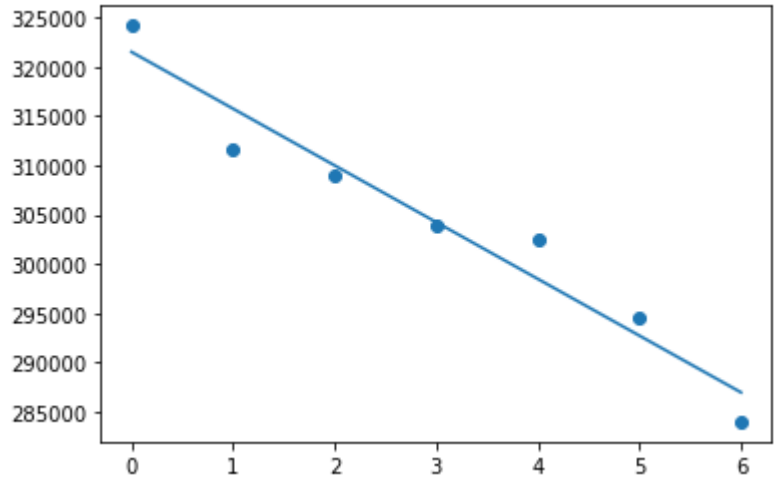


ScatterPlot

Nedenfor ses to scatterplots, hvor x-aksen repræsentere type af kriminalitet og y-aksen er antal begået kriminalitet. Her ses ingen lineær sammenhæng.



Nedenfor ses et scatterplot for antal kriminalitet begået på ugedage. Her ses det, at der er en lineær sammenhæng.



Ud fra disse visualiseringer kan vi konkludere at tyveri er den mest begået kriminalitet i San Francisco, og sandsynligheden for at støder på en kriminel er en smule højere på en fredag end på ugens øvrige dage.

Clustering

Clustering er en process, hvor man gruppere enheder sammen, som har fælles eller lignende egenskaber. Målet med en “unsupervised machine learning” -teknik er at prøve at finde grupper af data, som har lignende eller ens egenskaber og herefter at sætte dem sammen i en gruppe.

K-mean clustering og Hirarchical clustering er de mest kendte algoritmer inde for clustering, hvilket også er de 2 typer, som vi har taget udgangspunkt i.

K-mean fremgangsmåde

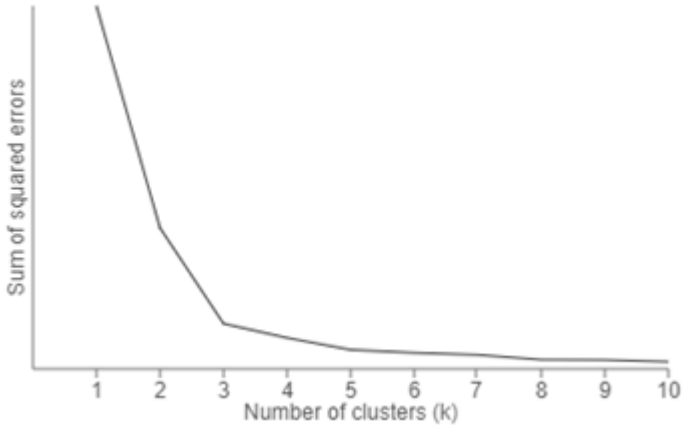
1. K-mean clustering fungere på den måde, at det starter med at tage imod K som input. K er repræsenteret, ved hvor mange klynger eller Clusters der forekommer. K centroide vil være placeret tilfældige steder i vores rum.

2. Ved brug af euklids distance mellem datapunkterne og centroiderne, tildele hvert datapunkt den klynge der er tættest på.

3. Herefter genberegn klynge-centrerne som et gennemsnit af de datapunkter, det er blevet tilknyttet.

4. Punkt 2 og 3 gentages indtil der er ikke forekommer flere ændringer.

Til Det første punkt af k-mean clustering kan man finde K ved brug af en metode kaldet “albue/elbow” metoden. Albue metoden bruges til at finde det optimalte antal cluster til algoritmen. Et eksempel på hvordan sådanne en beregnings udfald kunne se ud, kan ses på billedet nedenfor. Det viser at der er et naturligt knæk ved tallet 3, som indikeret at den marginale forskel vi betyde mindre efter 3 “clusters”.



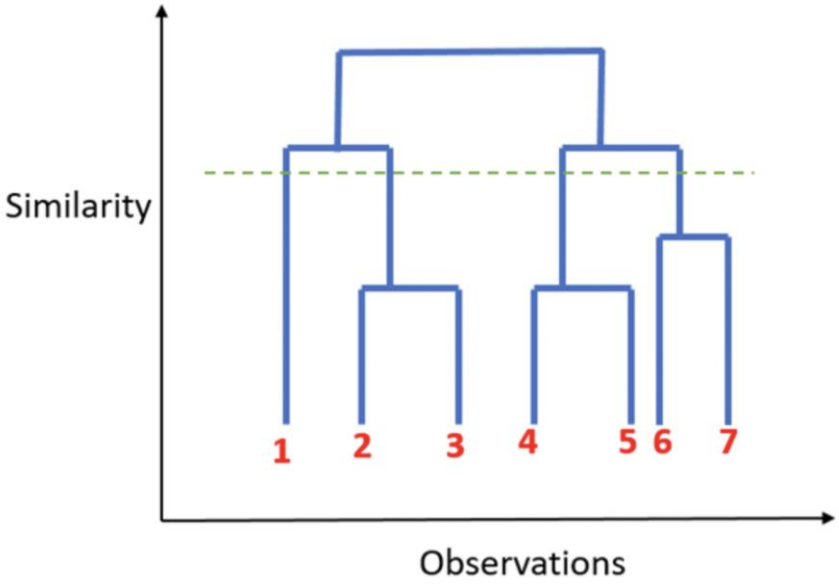
Hierarchical Clustering

Hirakist clustering algoritmen er en smule anderledes bygget op, den starter med at tildele alle data point, som sin egen cluster/klynge. Den bliver opbygget hirakisk, som navnet også indikere. Anden del af metoden kombinere de to tætteste punkter og samler dem som én klynge/cluster.

Hierarchical Clustering fremgangsmåde

1. Hvert datapunkt får deres eget cluster.
2. De nærmest liggende cluster/klynger findes ved brug af euclids distance og samles til én klynge/cluster.
3. Distancen mellem de to nærmeste klynger/cluster beregnes og kombineres. Dette gøres indtil alle klynger/cluster er én stor klynge/cluster

Der er ikke en fast regel til at fastsætte antal cluster/klynger. Ses der på billedet nedenfor, vil der i det tilfælde blive inddelt fire cluster/klynger, da der er fire selvstående grupper. Det kan også fastsættes, ved at se på hvor mange linjer der bliver brudt på tværs.



Beskrivelse af datasættet

Datasættet vi har benyttet til at lave vores clustering eksempler, er et datasæt fra machine learning repository[1]. Datasættet er baseret på en masse unikke brugere, der har oplevet 10 forskellige kategorier og givet de pågældende oplevelser en

[1] https://archive.ics.uci.edu/ml/datasets/Travel+Reviews

karakter fra 0 til 4. Karakteren 0 svarer til en dårlig oplevelse og Karakteren 4 svarer til en fantastisk oplevelse, 2 er en middel oplevelse.
Datasættet indeholder 980 rækker og 11 kolonner.

```
In [2]: data = pd.read_csv("tripadvisor_review.csv")
        data
```

Out[2]:

	User ID	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8	Category 9	Category 10
0	User 1	0.93	1.80	2.29	0.62	0.80	2.42	3.19	2.79	1.82	2.42
1	User 2	1.02	2.20	2.66	0.64	1.42	3.18	3.21	2.63	1.86	2.32

Variable information

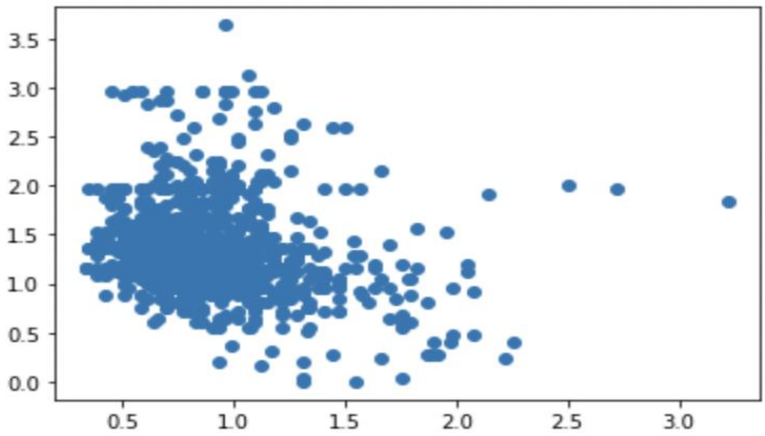
Attribute 1 : Unique user id
Attribute 2 : Average user feedback on art galleries
Attribute 3 : Average user feedback on dance clubs
Attribute 4 : Average user feedback on juice bars
Attribute 5 : Average user feedback on restaurants
Attribute 6 : Average user feedback on museums
Attribute 7 : Average user feedback on resorts
Attribute 8 : Average user feedback on parks/picnic spots
Attribute 9 : Average user feedback on beaches
Attribute 10 : Average user feedback on theaters
Attribute 11 : Average user feedback on religious institutions

Vores første undersøgelse af dette datasæt gik ud på at se, om der kunne skabes en correlation for hvordan brugere gav deres karakter baseret på to aktiviteter.

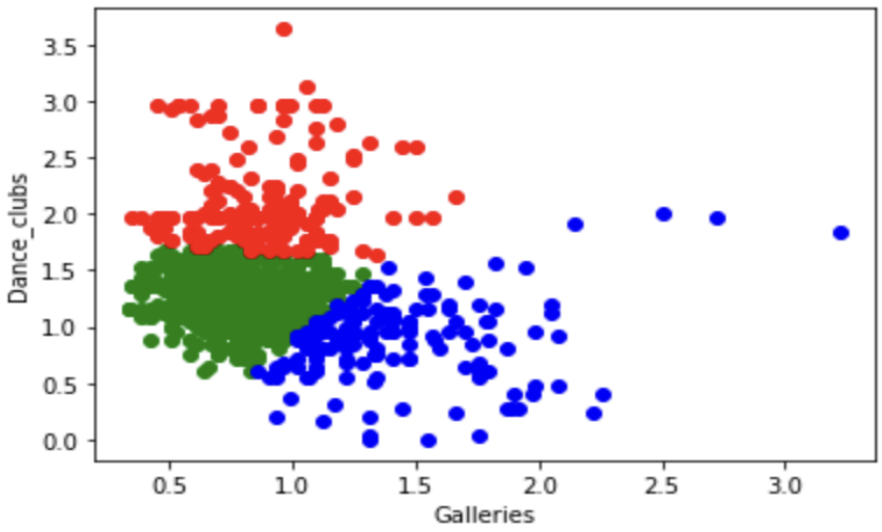
Analyse

Vores forsøg på at skabe et overblik over de to kategorier, som hedder Galleries og Dance_clubs, ses på billedet nedenfor. Det ligner umiddelbart én stor samling af karaktere.

```
In [4]: plt.scatter(data['Galleries'],data['Dance_clubs'])
Out[4]: <matplotlib.collections.PathCollection at 0x1c4765ddf40>
```

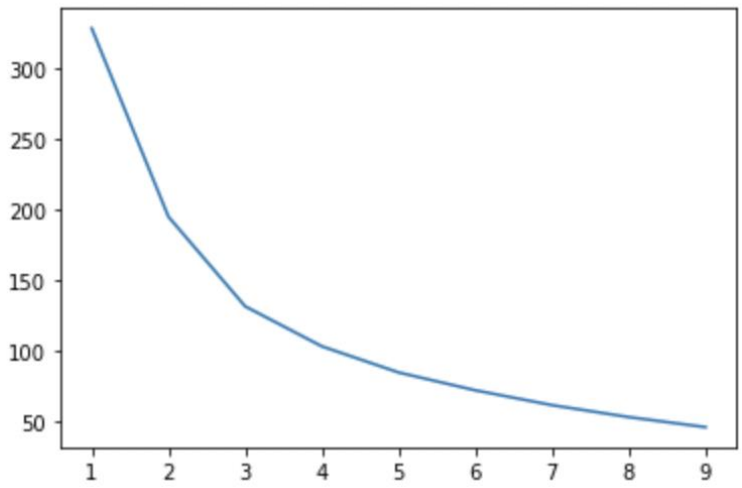


Vi vil benytte k-mean modellen, til forhåbentlig at opnå nogle clusters i vores datasæt. Ved brug af k-mean modellen er vi starter ud med at gætte på at der er 3 clusters. Efter at have fået inddelt datapunkterne i clusters, har vi valgt at farvelægge de 3 clusters og printe resultatet.
Ud fra figuren nedenfor ses det hvordan k-mean har inddelt de 3 kategorier. Størstedelen af alle brugere har givet en mindre tilfredsstillende karakter, men det ses at brugere der har givet en høj karakter sjældent giver en tilsvarende høj karakter i den modsatte kategori.
Vi kan altså umiddelbart konkluderer at mennesker der er tilfredse med danseklubber, ikke er glade for oplevelser der involverer gallerier (en karakter givet bpå over to).



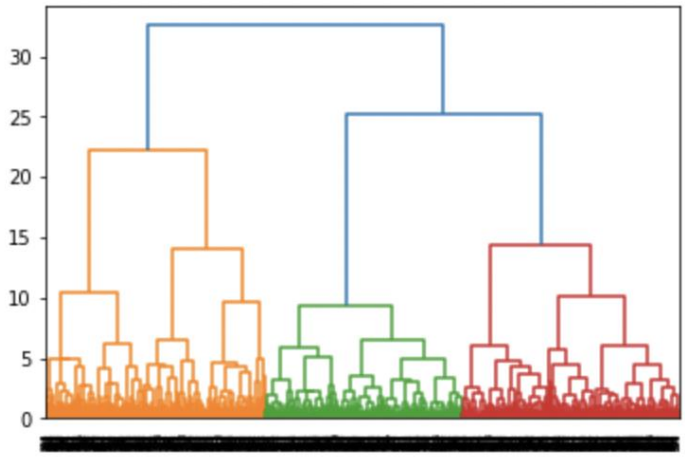
For at finde det optimale antal klynger/clusters, kan “albue metoden” benyttes. Her bruges SSE “Sum of the squared difference between each observation and its groups mean”. Billedet nedenfor indikere at der er mindre betydelige forskelle efter 3 klynger. Det er indikeret ved at grafen flades ud efterfølgende.

```
Out[39]: [<matplotlib.lines.Line2D at 0x7f9d68147370>]
```



Ud fra samme data, kan vi udføre et dendrogram. Dendrogrammet er opbygget fra et bottom up koncept, hvor alle datapunkter starter som individuelle klynger, og bliver dannet med nærmeste end til hele alt er forbundet i toppen af dendrogrammet. Dendrogrammet kan bruges til at skabe et andet visuelt perspektiv på ens datasæt.

```
dendogram = sch.dendrogram(sch.linkage(data, method ='ward'))
```



Vi har foretaget samme øvelse med et andet datasæt for at se om resultatet er konsekvent med vores metodevalg.

Beskrivelse af datasættet Clustering 2.

Det andet datasæt vi har brugt har oplyst persons alder og hvor meget de tjener. Derfor kunne det være interessant og se om der forekommer clusters i forbindelse med alder og indkomst?

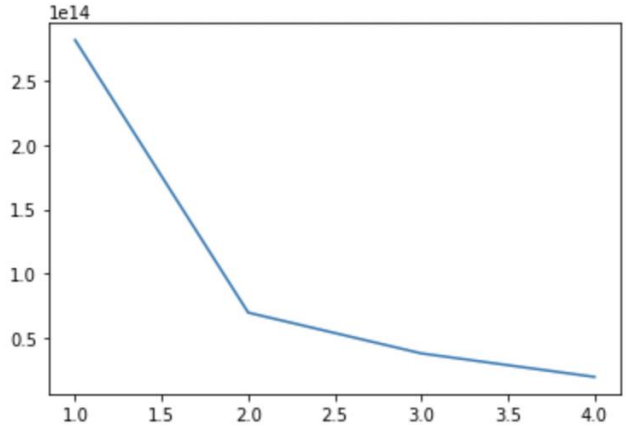
Out[46]:

	Unnamed: 0	Age	Employment Type	GraduateOrNot	AnnualIncome	FamilyMembers	ChronicDiseases	FrequentFlyer	EverTravelledAbroad	TravelInsurance
0	0	31	Government Sector	Yes	400000	6	1	No	No	0
1	1	31	Private Sector/Self Employed	Yes	1250000	7	0	No	No	0

Det andet datasæt har et meget tydeligt albue knæk der indikere det optimale antal kluster der findes ved variablerne age og income, er 2.

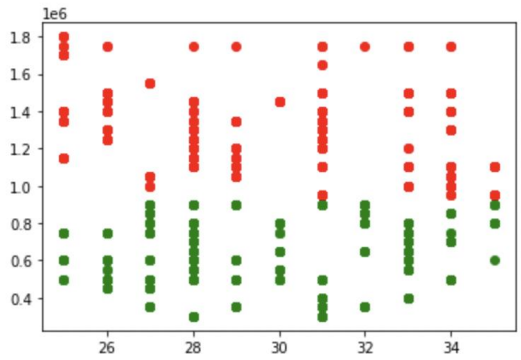
```
#plotter
plt.plot(k_rng,sse)
```

```
[<matplotlib.lines.Line2D at 0x7ff75859c430>]
```



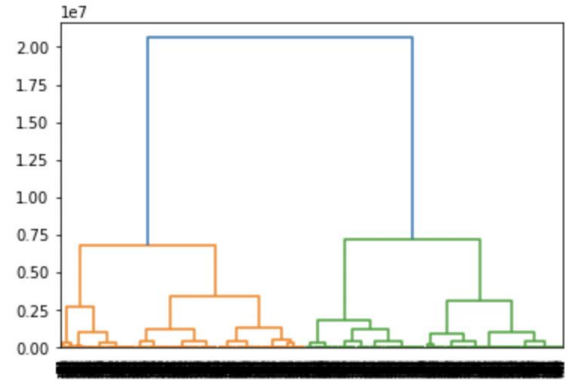
De 2 clustere visualiseret. det ses ud fra billedet nedenfor at opdelingen ligger horizontal, hvilket ikke er meget sigende om selve alder inddelingen. Det indikere mere en inddeling af rig fattig.

```
<matplotlib.collections.PathCollection at 0x7ff74ba347f0>
```



Der ses på dendogrammet at de også er indelt i 2 nærmeste kategorier.

```
dendogram = sch.dendrogram(sch.linkage(df, method ='ward'))
```



Klassifikation

Når vi kigger på området klassifikation inden for machine learning, kan vi ikke undgå at nævne at dette emne omhandler to primære typer af machine learning modeller. Der er machine learning for regression og så er der for klassifikation.

Regression bruges til at forudse resultater der er kontinuerlige, og er ikke begrænset af prædetermineret labels.

Et eksempel på regression kunne være:
At prøve at forudsige, hvad en persons højde eller indkomst er, baseret på to andre variabler. De to variabler kunne være alder og den by de bor i.
Modellen ville også kunne undersøge, hvor meget omsætning en kunde vil generere.

Klassifikation bruges til at forudse om udfaldet tilhører en prædetermineret "label/kategori" og i mange tilfælde bruges kun to mulige udfald 0 eller 1. Det er også kaldet binært klassifikation.

Et eksempel på dette kunne være:
Vil det regne eller ej.

Der findes også multi klassifikation, hvilket vil sige at der er mere en to mulige udfald.
Det kunne fx handle om, om et billede viser en kat, en hund eller en fugl. - Tre forskellige udfald.

Der er ikke én metode/algoritme der virker til alle datasæt.
Ses der på billedet nedenfor, kan man bruge den som en vejledende oversigt, for en foretrukken algoritme.

Logistic Regression vs Random Forest Classifier

Type of training set	Preferred Algorithm
Linearly Separable	Logistic Regression
Multi-Dimensional	Random Forest Classifier
Categorical Variable	Random Forest Classifier
Continuous Variable	Logistic Regression

Beskrivelse af datasæt

Vi har valgt at bruge “Titanic” datasæt fra pydataset, som er en package fra python. Titanic datasæt indeholder fire variabler, som hhv. er “class”, “age”, “sex” og “survived”. Class indeholder tre forskellige klasser, som er 1.klasse, 2.klasse og 3.klasse. Age bliver inddelt i børn eller voksen, og survived er om personen har overlevet.

Analyse

I analyseafsnit har vi lavet en “random forest classifier” og en deep learning.

Random forest classifier
I dette afsnit vil vi finde ud af hvilke faktorer der er med til at skabe større chance for overlevelse.

Før vi går i gang med modellen, har vi omdannet al tekst til tal, hvilket kan ses på billedet nedenfor.

```
In [191]: #Omdanner til tal
          df = pd.get_dummies(titanic,drop_first=True)
          df
```

Out[191]:

	class_2nd class	class_3rd class	age_child	sex_women	survived_yes
1	0	0	0	0	1
2	0	0	0	0	1
3	0	0	0	0	1
4	0	0	0	0	1
5	0	0	0	0	1
...	...	...	...	...	...
1312	0	1	1	1	0
1313	0	1	1	1	0
1314	0	1	1	1	0
1315	0	1	1	1	0
1316	0	1	1	1	0

1316 rows × 5 columns

Derefter bruger vi train_test_split til at splitte datasæt i train og test.

```
In [153]: #spliter datasæt til traningsdatasæt og testdatasæt
          X_train, X_test, Y_train, Y_test = train_test_split(df.drop('survived_yes', axis=1),df['survived_yes'],test_size=0.1,random_state
```

Vi laver en random forestclassifer model, og fitter vores trainings-værdi ind.

```
In [183]: #Laver model og træne
          model = RandomForestClassifier()
          model.fit(X_train,Y_train)
```

Herefter bruger vi følgende kommandoerne til at analysere vores model.

- model.predict(x_test)
- model.predict_proba(x_test)
- model.score(x_test,y_test)

Nedenfor ses personernes sandsynlighed for at overleve, hvor venstre side er sandsynligheden for at man ikke overlever, og højre side er sandsynligheden for at overleve.

```
In [187]: #Sandsynligheden for at man overlever
          model.predict_proba(X_test)

Out[187]: array([[0.        , 1.        ],
                 [0.84055458, 0.15944542],
                 [0.92732007, 0.07267993],
                 [0.84055458, 0.15944542],
                 [0.66118717, 0.33881283],
                 [0.84055458, 0.15944542],
                 [0.84055458, 0.15944542],
                 [0.54503777, 0.45496223],
                 [0.54503777, 0.45496223],
                 [0.66118717, 0.33881283],
                 [0.14757156, 0.85242844],
                 [0.84055458, 0.15944542],
                 [0.92732007, 0.07267993],
                 [0.84055458, 0.15944542],
                 [0.84055458, 0.15944542],
                 [0.84055458, 0.15944542],
                 [0.84055458, 0.15944542],
                 [0.84055458, 0.15944542],
                 [0.92732007, 0.07267993],
```

Med de sandsynligheder kan vi sammenligne med nedenstående tabel, og finde ud af hvilke faktorer der skaber større chance for at overleve.

```
In [185]: X_test
```

Out[185]:

	class_2nd class	class_3rd class	age_child	sex_women
599	1	0	1	1
855	0	1	0	0
371	1	0	0	0
899	0	1	0	0
93	0	0	0	0
1068	0	1	0	0
717	0	1	0	0
1138	0	1	0	1
1086	0	1	0	1
148	0	0	0	0
556	1	0	0	1
877	0	1	0	0

Til sidst har vi tjekket hvor præcis vores model er. Vores resultat er 0.82, hvilket er ret højt, dog er den ikke helt præcist.

```
In [189]: #Tjekker hvor godt vores model er fra 0-1
          model.score(X_test,Y_test)

Out[189]: 0.8181818181818182
```

På nedenstående billeder ses det, at en kvinde eller en pige fra første klasse vil overleve, mens en mand fra første klasse vil ikke.

```
In [196]: #Tjekker om en kvinde fra første klasse vil overleve.
          model.predict(np.array([[0,0,0,1]]))[0]

Out[196]: 1
```

```
          model.predict(np.array([[0,0,1,0]]))[0]

Out[202]: 1
```

```
          model.predict(np.array([[0,0,0,0]]))[0]

Out[201]: 0
```

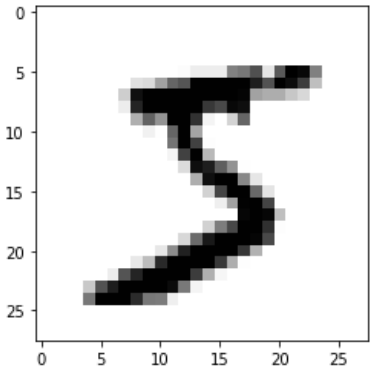
Ud fra dette kan vi konkludere, at køn er den mest betydende faktorer på om man overlevede under katastrofen.

Deep learning

Vi benytter Keras datasæt, som indeholder en masse håndskrevet billeder i størrelsesformattet 28x28 pixels. Indeller datasættet i train og test, som ses på billedet nedenfor.

```
In [174]: mnist = tf.keras.datasets.mnist   #Load en 28x28 billede
          (x_train,y_train),(x_test,y_test) = mnist.load_data()
```

```
In [154]: plt.imshow(x_train[0], cmap = plt.cm.binary)
          plt.show()
```



Hvor efter vi har lavet en model og trænet den, som ses på billedet nedenfor, hvor man kan se at “accuracy” stiger efter hver training.

```
In [161]: model = tf.keras.models.Sequential()
          model.add(tf.keras.layers.Flatten())
          model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))
          model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))
          model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax))
          model.compile(optimizer= 'adam',
                        loss="sparse_categorical_crossentropy",
                        metrics=['accuracy'])
          model.fit(x_train,y_train,epochs=3)

          Epoch 1/3
          1875/1875 [==============================] - 5s 2ms/step - loss: 1.7849 - accur
          acy: 0.8798
          Epoch 2/3
          1875/1875 [==============================] - 4s 2ms/step - loss: 0.3320 - accur
          acy: 0.9345
          Epoch 3/3
          1875/1875 [==============================] - 4s 2ms/step - loss: 0.2093 - accur
          acy: 0.9477
```

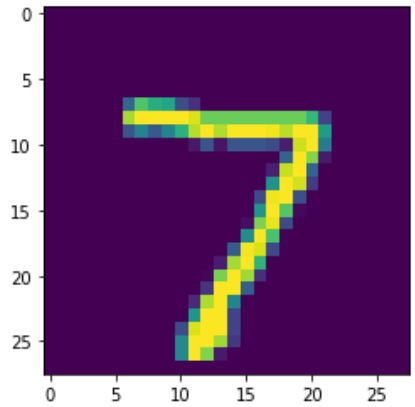
Til sidst kan vi bruge predict til at finde ud af hvilke tal er det egentlig test[0;9] indeholder. På billedet nedenfor bruger vi vores model, og predictions[0] til at finde ud af hvilket cifre den indeholder. Vi har fået den til at være 7, hvorefter vi kan plotte det og konkludere om vores prediction er rigtigt eller falsk.

```
In [183]: print(np.argmax(predictions[0]))

          7
```

```
In [184]: plt.imshow(x_test[0])

Out[184]: <matplotlib.image.AxesImage at 0x24890779a90>
```



Sentiment

I sentiment afsnit har vi brugt TextBlob til at analysere ordets polaritet og subjektivitet, hvor polaritet har en interval på [-1.0;1.0] og subjektivitet har en interval på [0.0;1.0].

Nedenfor ses eksempel af nogle adjektiver som “Great”. Hermed kan man se at polarity er væsentligt højt, da det er et positivt ord. Hvis man tilføjer ‘Not” så bliver polarity til -0.4, hvilke er negativt. For subjektivitet er de allesammen høje, da det er ens egen mening.

```
In [89]: TextBlob("Great").sentiment
Out[89]: Sentiment(polarity=0.8, subjectivity=0.75)

In [90]: TextBlob("Not Great").sentiment
Out[90]: Sentiment(polarity=-0.4, subjectivity=0.75)

In [91]: TextBlob("Very Great").sentiment
Out[91]: Sentiment(polarity=1.0, subjectivity=0.9750000000000001)

In [29]: TextBlob("i am great").sentiment
Out[29]: Sentiment(polarity=0.8, subjectivity=0.75)
```

Herefter har vi brugt en datasæt som handler om hotels anmeldelser fra tripadvisor, hvor vi analysere anmeldelsers positivitet og subjektivitet.
Der ses 10 eksempler nedenfor, hvor de fleste polaritet er under 0.5, hvilke betyder at kommentarerne har været meget på den negative side.
I forhold til subjektivitet er de fleste til gengæld over 0.5, hvilke giver også meget god mening, da anmeldelser handler meget om sin egen holdning.

```
In [28]: df.sample(10)
```

Out[28]:

	Review	Rating	polarity	subjectivity
18556	just returned majestic just returned somewhat ...	3	0.221090	0.578750
7101	highly recommend, review westchester ny time, ...	5	0.210894	0.504921
14929	loved hotel stayed new year hotel really fab g...	5	0.401824	0.656753
16039	good choice wanted hotel close relations flat ...	4	0.194744	0.504002
5980	n't worry old reviews just returned staying hi...	4	0.110608	0.472735
763	best marriotts stayed, check checkout staff wo...	5	0.744444	0.727778
2516	beautiful el convento having read reviews deci...	4	0.190212	0.517460
10135	perfect place stay staff hotel gave warm welco...	5	0.190727	0.472400
4897	hotel artus hard beat stayed twice 1996 recent...	5	0.272976	0.618036
7557	great hotel awesome experience, family stayed ...	5	0.500303	0.605455

Reference

https://data.sfgov.org/browse?q=incident%20report&sortBy=relevance
https://data.sfgov.org/Public-Safety/Police-Department-Incident-Reports-Historical-2003/tmnf-yvry

https://scikit-learn.org/stable/auto_examples/index.html#cluster-examples

https://archive.ics.uci.edu/ml/datasets.php?format=&task=clu&att=&area=&numAtt=&numIns=&type=&sort=dateDown&view=table (

https://archive.ics.uci.edu/ml/datasets/Travel+Reviews (Dataset til clustering)
https://www.kaggle.com/datasets/andrewmvd/trip-advisor-hotel-reviews (Dataset til sentiment)

Logistic Regression vs Random Forest Classifier