Machine learning

MNIST



Andreas Wendel

EC Utbildning

ML- Kunskapskontroll 2

2024-03

# Abstract

“AI is on the rise” sounds like a sci-fiction plot but each day that statement is becoming more and more of a reality. AI is growing in an exponential pace and in this paper, we will research some of the capabilities. When researching AI, you will most likely come across Machine learning and predictions and in this paper, we will create a project where we use images and try and train a model which will then predict that image. This is the first step to AI being able to se what we humans se.

# Förkortningar och Begrepp

AI:\_\_\_\_\_\_\_\_\_ Artificiell intelligens

ML: \_\_\_\_\_\_\_\_ Machine learning

Extra Trees: \_\_ Extremely Randomized Trees

SVM: \_\_\_\_\_\_\_\_Support Vector Machine

SVMC: \_\_\_\_\_\_\_ Support Vector Machine Classifier

LR: \_\_\_\_\_\_\_\_\_\_ Logistic Regression

MBKM: \_\_\_\_\_\_ MiniBatchKmeans

Ca: \_\_\_\_\_\_\_\_\_\_ Cirka

TP: \_\_\_\_\_\_\_\_\_\_ True Positive

TN: \_\_\_\_\_\_\_\_\_\_ True Negative

FP: \_\_\_\_\_\_\_\_\_\_ False Positive

FN: \_\_\_\_\_\_\_\_\_\_ False Negative

Innehållsförteckning

[Abstract 2](#_Toc162021359)

[Förkortningar och Begrepp 3](#_Toc162021360)

[1 Inledning 1](#_Toc162021361)

[1.1 Frågeställningar 1](#_Toc162021362)

[2 Teori 2](#_Toc162021363)

[2.1 Dataset 2](#_Toc162021364)

[2.1.1 Hur mycket data skall modellen hantera 2](#_Toc162021365)

[2.1.2 Ska man skala ner datan 2](#_Toc162021366)

[2.2 Välja modeller 3](#_Toc162021367)

[2.2.1 Supervised learning 3](#_Toc162021368)

[2.2.2 Unsupervised learning 6](#_Toc162021369)

[3 Metod 8](#_Toc162021370)

[3.1 Hantering av data 8](#_Toc162021371)

[3.2 validering processen 9](#_Toc162021372)

[3.2.1 Välja modell 9](#_Toc162021373)

[3.3 Evaluerings process 16](#_Toc162021374)

[3.4 Prediktera riktiga bilder 17](#_Toc162021375)

[3.4.1 Konvertera bilder 17](#_Toc162021376)

[3.5 Webbapplikation (streamlit) 17](#_Toc162021377)

[4 Resultat och Diskussion 18](#_Toc162021378)

[4.1 Data 18](#_Toc162021379)

[4.2 Validering 18](#_Toc162021380)

[4.2.1 Kmeans 18](#_Toc162021381)

[4.3 Prediktering och Slutlig modell 19](#_Toc162021382)

[5 Slutsatser 20](#_Toc162021383)

[6 Teoretiska frågor 21](#_Toc162021384)

[7 Självutvärdering 23](#_Toc162021385)

[Källförteckning 24](#_Toc162021386)

# Inledning

AI är ett område som kraftig växer fram i dagens samhälle. Sedan OpenAi ’s Chatgpt så har populariteten av AI växt något enormt, Man ser företag som har hand om datorkraft och utveckling av AI bli evaluerade till miljarder av dollar, nyheter och inlägg på sociala medier om AI. AI är ett brett område och djupare in så handlar det om Maskin inlärning alltså att lära en maskin(datorn) något. ML har dock funnits länge hela vägen sedan 1950 när Arthur Samuel uppfann ett program som kunde kalkylera chansen för vinst i spelet Dam för båda sidorna. I dag finns det många tillämpningar till ML och ett av dem är bildigenkänning. En ML modell har möjligheten att identifiera olika objekt genom att kolla på en bild, man kan exempelvis identifiera sjukdomar genom att mata in bilder, eller kategorisera in olika djur som datorn ser på bilden.

ML modeller tränas på data som innehåller oberoende variabler också kallat features. Dessa features är variabler som säger något om data och till slut resulterar till vår beroende variable också kallat label. Låt säga att man vill beräkna lön. Då är lön våran label och features kan vara information som ålder, utbildning, erfarenhet osv.

I denna rapport kommer man att bygga och undersöka en ML modell som kommer kunna prediktera handskrivna siffror.

Detta kommer ske genom att Träna flera ML modeller på ett dataset som innehåller handskriva Siffror, modellerna kommer också att granskas och valideras. En slutgiltig modell kommer sedan att evalueras på egna handskrivna siffror som kommer att konvertera till data som modellen kan läsa.

## Frågeställningar

För att uppfylla detta syftet kommer vi att svara på följande frågor.

1. Dataset
   1. Hur mycket data skall modellen hantera
   2. Ska man skala ner datan
2. Välja modeller
   1. Supervised learning
      1. Logistic Regression
         1. Hyperparametrar
      2. Support Vector Classifier
         1. Hyperparametrar
      3. Extremely Randomized Trees
         1. Hyperparametrar
   2. Unsupervised learning
      1. Mini Batch KMeans
         1. Hyperparametrar
3. Prediktera Handskrivna
   1. konvertera bild till data
4. webbapplikation (Streamlit)

# Teori

## Dataset

Man kommer använda sig Sklearns MNIST dataset. Datasetet innehåller matriser som representerar bilder. Matrisen innehåller pixlar som är formaterad till en kolumn med 784 rader. Hela datasetet innehåller 70000 kolumner eller bilder. För få fram en bild av en kolumn kan du dra roten ur antalet rader (roten ur(784)=28) och få fram en area. Denna arean kommer då att representera en bild av med 28x28 pixlar och innehålla en siffra.

### Hur mycket data skall modellen hantera

I vanliga fall är det bra att träna modellen på ca 80% av datan och testa på resterande 20%. I fallet när man hanterar validering kan det va bra att minimera andelen tränings data för att minska träningstid i valideringsprocessen.

### Ska man skala ner datan

Maskin inlärningsmodeller hanterar ofta i majoriteten av tillfällen datan bättre om den är skalad. Att Skala data betyder i normal fallen att minimera bredden i data. I många fall kan man normalisera datan med en standard-scaler där man sätter μ=0 och σ=1( ) där och . Ett annat sätt att skala datan är att få matrisen att bara innehåller 1:or och 0:or och efter som att vi vet att varje pixel kan vara mellan 0-255 då kan man dividera varje pixel med 255

## Välja modeller

Att välja modell handlar om att anpassa modellen till ens problem. I detta fall handlar det om Bildigenkänning vilket vi kan klassificera som ett klassificerings problem. Många modeller kan hantera klassificerings problem och ingen modell är egentligen fel. Det leder till att man utför en validering på X antal modeller för att se vilken modell som anpassas bäst till att hantera problemet. Enligt en undersökning av Kaggle 2017 (se Figur 1) kan vi se att Logistic Regression används mest av alla följt av Random forest och Decision Tree. Man har valt att välja modeller som fungerar på olika sätt jämfört med varandra därför väljer man att inte bara ta dem top mest använda. För att diversifiera så har man valt att välja 3 Supervised learning modeller och 1 Unsupervised learning model.

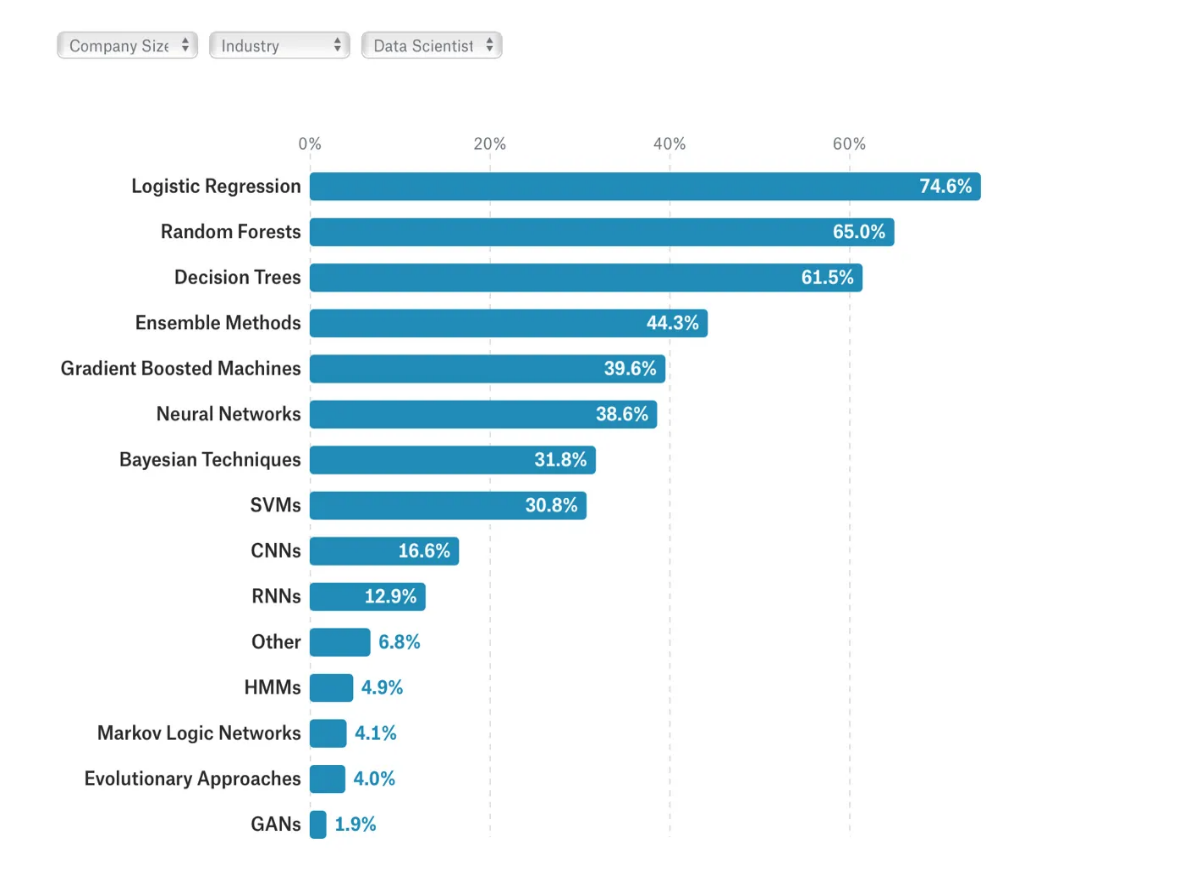
### Supervised learning

Supervised learning handlar om att man tränar modellen på data som innehåller som också innehåller labels, alltså rätt svar till varje datapunkt. Detta medför att modellen kan associera en data punkt till ett svar och sedan har en referera till den om det kommer upp liknande data.

#### Logistic Regression

Logistic Regression, den vanligaste modellen bland Data Scientist. Det är en binär klassificierare som uppskattar sannolikheten att en observation till hör en klass. I vårat fall beräknar den sannolikheten att datapunkt skall vara en siffra. Matematisk sätt vill vi räkna ut p(sannolikheten) för att en data punkt skall vara en siffra, där

##### Hyperparametrar

Olika modeller kan optimeras genom att ändra på parametrar på modellen. I Logistic Regressions modellen kan man ändar på ”penalty” som står för Regularization Penalty genom att ändra på penalty till L1 (Lasso) eller L2 (Ridge) så kan man stoppa modellen från att överfitta. C är även en parameter som hanterar Regulariserings styrka, Högre C minskar regulariseringen och vise versa. Modellen har även en parameter som kallas för Solver vilket bestämmer vilken optimerings algoritm modellen skall använda, olika Solver presterar bättre på vis data.

Figur 1

#### Extremely Randomized Trees

Extremely Randomized Trees eller Extra Trees är en ensemble learning method som tillhör familjen av beslutsträd. För att förstå Extra trees måste man först förstå vad Beslutsträd är och vad ensemble learning är. Besluts träd fungerar genom att skapa noder och lövnoder efter den har tränats på datan som sedan sprids ut som ett träd (se Figur 2) Så för varje nod går modellen igenom kriterier på nya datan och skickar sedan vidare datan till antigen nästa nod eller till slutligen en löv nod där den får en klassificering. Modellen innehåller också en ett mått på ”ren” het som heter Gini den koefficienten beräknar ”renheten” av en löv nod alltså andelen enskillda klassificerade data i lövnoden. Låg Gini betyder att vi har hög ”renhet” alltså att löv noden innehåller enbart en typ av klassificering

Generelt sätt är Ensamble learning är en metod där man kombinerar flera modeller intill ett system där majoriteten av predikterat resultat blir de slutgiltiga resultatet. Exemple modellerna röstar A:ja, B:nej, C:ja kommer Ensamble returnera Ja. Ensamble tränar även dessa modeller på olika varianter av Datan.

Egentligen så är Extra trees en Random Forest modell men den lägger med ytligare slumphet till varje konstruktion till av nod. Så vad är Random Forest? Random Forest Modell baseras på Beslutsträd Men för varje beslutsträd så slumpmässigt väljs features för varje lövnod och detta är för att försöka minska Gini koefficienten. Modellen tränas även på en ensamble metod där flera beslutsträd tränas enskilt på en del av data med återläggning detta kallas för bagging. Detta hjälper då att minska variansen genom att ta mellanvärdet av prediktioner från flera beslutsträd. Slutligen så kommer ensamble metoden ta in alla prediktioner från beslutsträd och rösta fram en prediktion.

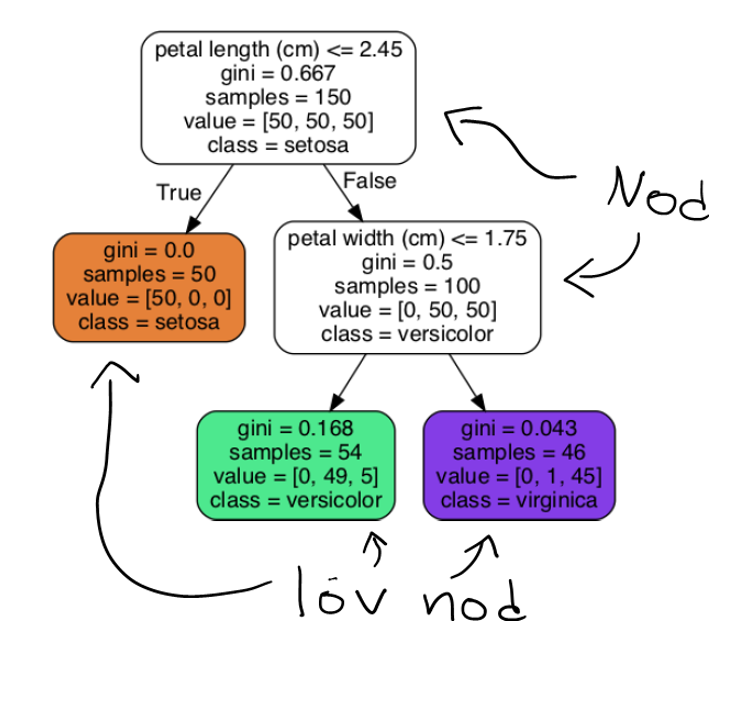
##### Hyperparametrar

Med Extra Trees finns det hyperparametrar som

N\_estimator: bestämmer antalet beslutsträd. Ökning va beslutsträd kan leda till bättre generalisering prestanda speciellt vid stora och komplex data men det kräver mera dator kraft.

Splitter: bestämmer strategin för bär man skall split vid noder.

Max\_depth: kan välja att begränsa hur långt ner ett beslutsträd skall gå. Kan va bra för att modellen inte skall överfittas.

Figur 2

#### Linear Support Vector Classifier

Linear Support vector Classifier kommer ifrån Support Vector Machine för klassificering.

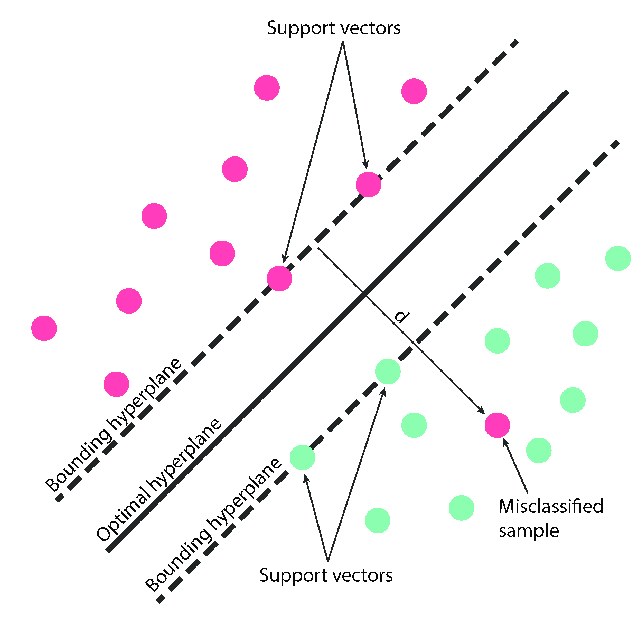
Modellen fungerar genom att försöka hitta hyperplan som bäst kan separera datan intill klassifikationer (Se Figur 3). Hyperplanen försöker skapa en sådan ”bred” vägg mellan klasser som den kan. Den sätter väggar mellan datapunkter som kallas för Support vectorer och dessa datapunkter används för att separera klasserna. Detta kan vara väldigt svårt för vissa datapunkter eftersom data sättas nära fel klasser, detta kan man lösa genom en metod som heter soft Margin classifier som hjälper till att regalera antalet ”outlier” i datan alltså data som tillåts vara inom vårat hyperplan. LInearSVC funkar likadant som SVM men skillanden är att den redan har valt Kernal funktion till en linjär funktion, detta är för att den enklare hanterar större dataset enklare. Kernal funktkion är ett trick som hjälper modellen att flytta upp data till en högre dimension så att det bli enklare för modellen att hitta ett hyperplan

Något att tillägga är att SVM kan fungera för både regression problem och klassificerings problem och funkar bäst med mindre dataset eftersom det kräver mycket datorkraft och tid att träna en sådan modell och det är därför väldigt viktigt att skala ner sin data så att modellen enklare kan tränas.

##### Hyperparametrar

LinearSVC har lite mindre hyperparametrar än SVC men dem kan hjälpa oss regularisera och optimera modellen ändå.

C: är en viktigt parameter som låter oss reglera så kallade margin violations, alltså datapunkter som kommer innanför våra support vektorer. En väldigt låg c kan hjälpa modellen att generalisera bättre.’



Figur 3

### Unsupervised learning

Unsupervised learning fungerar till skillnad från supervised learning utan en oberoende variabel(label) och där med jobbar med datan för att hitta egna strukturer och mönster utan någon hjälp eller vägvisning. Denna metoden är bra på att hitta egna lösningar eller gömda klassificeringar med hjälp av modeller.

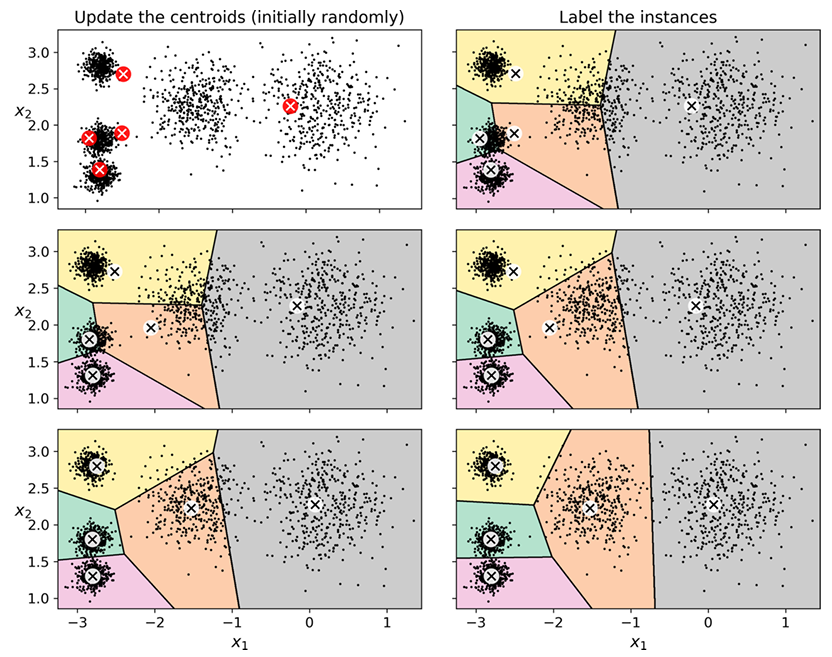
#### Mini Batch KMeans

MIniBatchKmeans är exakt lika dan som Kmeans men den gör det enklare och snabbare att träna en modell då den slumpmässigt väljer bitar(batches) från datan och träna på.

Kmeans är en unsupervised learning modell som jobbar genom att bygga kluster av data med ett förbestämt antal kluster. Data är utsprid och modellen jobbar i iterationer för att sätta ut så kallade centroider vars jobb är att hitta mitten av den klumpade datan (se Figur 4). Där efter kan modellen dela in centroiderna i kluster och separera datan in till kluster.

##### Inertia och Hyperparametrar

Att reglera en Unsupervised learning modell är svårt eftersom modellen inte tränas med en label data, detta göra att det egentligen inte finns något fel i modellens prediktioner. Dock så finns det en matematisk metod där vi kan kalkylera avståndet mellan en datapunkt och dess centroid. Detta heter Inertia och där beräknar vi medelvärdet av alla avstånd mellan en datapunkt och dess närmaste centroid. Detta medför dock problemet att kommer det vara bättre att öka antalet kluster efter som då kommer det matematiskt sätt minska medelvärdet av avstånd. Jobbet här är då att hitta en bra plats där inertia sluta minska kraftigt med antalet kluster.

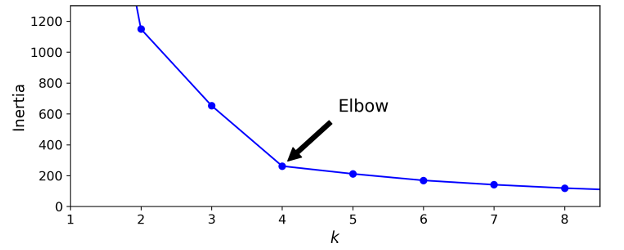
Man kan exemple vis kolla på hur inerta minska och hitta en ”armbåge” där minskningen signifikant minskar (se Figur 5) 

Figur 4

Om man kollar på hyperparametrana för Kmeans så har vi

N\_cluster: här väljer man antal kluster.

Init: Init eller initialization method specificerar metoden för hur man skall på börja kluster centroids instancieringen



Figur 5: notera armbågen vid kluster = 4, men om vi kollar på figur 2.3 så ser det bättre ut med 5 kluster.

# Metod

Allt kommer att programmeras i Python och man kommer även använda sig utav python paket så som sklearn, numpy, pandas och mera.

## Hantering av data

Man har valt att använda sig av MNIST datasetet som erbjuds av Scikit-Learns paket. Detta dataset kommer i form av en stor matris som innehåller represenationer av 28x28 bilder. Bilderna är i RBG form vilket betyder att dem har ett värde mellan 0-255 där värdet representerar en färg.

Man har valt att försöka skala ner datan från 255 värden till något mer lätt hanterat. Det finns 2 alternativ antigen kör man datan till värden mellan [0,1] väldigt enkelt matematiskt där . Det andra sätter handlar om att normalisera datan genom att sätta μ=0 och σ=1, (läs 2.1.2 sida 2 för ekvation). Så för att välja metod har man gjort en liten validering process där man tagit en klassificeringsmodell i detta fall Logistic Regression eftersom den är simple och väldigt populär bland andra data scientist, och där med tränat modellen på olika typer av data. Man tränade 3 modeller en utan någon skalning av datan, en med simple konvertering av pixel/255 och den sista med normalisering av data.

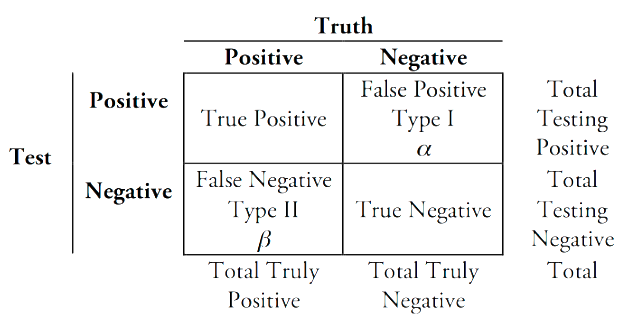
Datan har innehållit dem första 10000 bilderna från hela datasetet och för att evaluera kommer modellen Prediktera på dem nästa 1000 bilderna från datasetet. Den modell med bäst resultat kommer man att fortsätta använda. Modellen kommer få ut ett accuracy score från en simpel ekvation där accuracy score är . Det finns även ett annat sätt att få en score vilket är F1 Score där man kollar på en så kallad confusion matrix (se Figur 6). F1 Score kalkyleras med ekvationen Där och denna kalkylering kommer från Sklearn och ger oss en till parameter då vårat problem innehåller en multiklassificering, detta ger en möjligheten att välja mellan Micro, Macro, weighted och binär. Enkelt sätt så är Micro samma sak som vår andra score och macro kommer att kalkylera våra F1 score utan att aggregera några typer av viktningar till klasser. Weigther kalkylerar alla klassers F1 score individuellt och adderar tillsammans med viktning. Binär är för binär klassificering (se ii).

Vi kan också kolla på hur data ser ut i bildform efter vi har konverterat den (se Figur 7). Man ser en signifikant skillnad på bilden där datan är normaliserad, siffran ser skuggad ut och man får lite ”oljud” runt om kring. Detta kan både vara bra och dåligt. Man skulle kunna argumentera att den dåliga bilden kommer förstöra för modellen men man kan också säga att den kommer hjälpa modellen att anpassa sig för ”oljud” i framtida predikterade bilder.

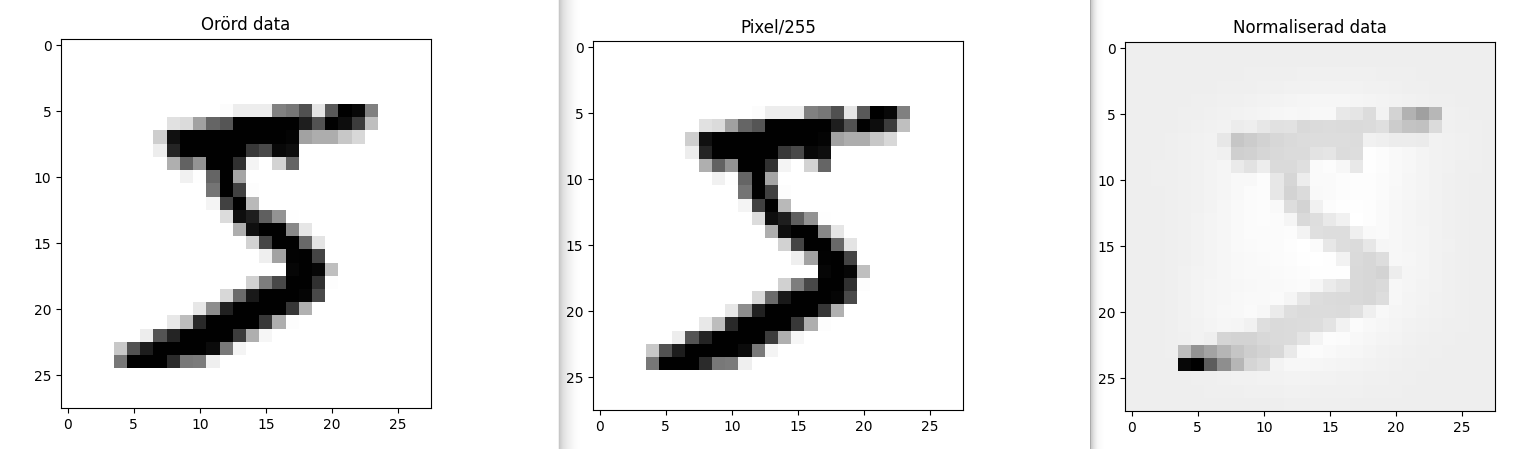
Om vi jämför resultaten så kan vi enkelt se att den enkel skalning på pixel/255 ger oss bäst resultat (se Tabell 1) så man väljer att fortsätta använda den typen av skalning. Man kommer i försättning att referera data skalad genom pixel/255 med simpelt skalade data.

Tabell 1

|  |  |  |  |
| --- | --- | --- | --- |
| Data Typ | Orörd Data | Simpelt skalad data | Normaliserad data |
| Accuracy Score | 0,887 | 0,932 | 0,907 |
| F1Score | 0,8852 | 0,9307 | 0,9057 |



Figur 6



Figur 7: Notera att det inte är någon skillnad från en orörda datan och den simpelt skalade data men när datan blir normaliserad blir det nästintill svårt att se siffran.

## validering processen

När man nu har skalat data kan man fortsätta med valideringsprocessen där man kommer att träna och välja den slutgiltiga modell.

Man börjar med att utöka våra datan, nu använder man 30000 bilder där dem första 20000 kommer att vara tränings data och de resterande 10000 vara våra validerings data som vi kommer att testa modellerna på.

### Välja modell

Här kommer man att välja 4 modeller varav 1 är en unsupervised learning modell, Man vill samt också att modellerna ska vara unika utifrån varanda så Random forest och beslutträd kommer ej vara ett bra val.

Om man referera igen till Kaggles Survay (se i eller Figur 2) så kan vi handplocka några populära modeller. Utav dessa så har man valt Extra Trees, SVC eller LinearSVC, man fortsätter att jobba med Logistic Regression och som unsupervised modell har man tagit Kmeans.

Dessa modeller kommer att tränas och smått optimeras för att rättvist få en stabil valideringsprocess. Man har valt att ta hjälp ifrån GridsearchCV som är en algoritm som tar in hyperparametrar och modeller och sedan tränar modellen på olika bitar av datan för att sedan få fram information utifrån en score som är liknande accuracy score, om vilka parametrar som har presterat bäst.

Man har dock valt att begränsa antalet hyperparametrar man testar eftersom det är väldigt kraft och tids krävande Man har också valt att alla modeller skall tränas med 3 folds för att bibehålla rättvis validering och man har också använt sig av en global parameter Random\_state=42 vilket ger alla slumpmässiga delar kommer börja med Seed=42, detta medför att vi får samma resultat om vi måste göra om någon process.

#### Logstic Regression

Vi har tidigare använt denna modellen för att evaluera datan och eftersom modellen gav en score runt ca 90% så har man valt att fortsätta med den modellen och smått optimera den med GridsearchCV.

Som hyperparameter har man valt att ändra bara på parametern Solver som ändra optimiserings algoritmen för modellen (Solver = [newton—cholesky,lbfgs,sag]). Efter som man hanterar ett multiclass problem så är dem här 3 bra enligt Sklearn (se iii)

Dessa resultat får man dock och vi kan se att Sag algoritmen get oss bäst resultat (se Tabell 2). Vi märker att de inte är större skillnad till score och det finns möjlighet till optimisering om man skulle välja denna modellen. Vi ser dock att det bästa resultatet tog med marginal mest tid att träna

Tabell 2

|  |  |  |  |
| --- | --- | --- | --- |
| Rank | Parameter | Mean fit time (s) | Mean-test-score |
| 1 | Sag | 92.59 | 0.9061 |
| 2 | Lbfgs | 15.06 | 0.9058 |
| 3 | Newton-cholesky | 26.9 | 0.9054 |

#### Extra Trees

Denna modellen kommer ifrån random forest som också är en väldigt populär modell. Man har valt Extra Trees istället för Random Forest för denna modellan skall med hjälp av den extra slummässigheten hjälpa modellen att inte bli alldeles för överfittad.

Som hyperparametrar kommer man att testa olika n\_estimators som är antalet beslutsträd som modellen kommer att skapa. Dessutom kommer man även ha en till parameter där vi testar att har ”None” max\_depth och en begränsad max\_depth på 5. När max\_depth är None så kommer träden att växa till alla lövnoder är helt rena eller tills max\_split är nådda. Detta kan medföra en modell som blir väldigt stor.

Våra parametra är (n\_estimater=[100,200,300,400,500,600] och Max\_depth=[None,5])

Vi kan se resultaten i tabellen (se Tabell 3) och ser att vi har 3 resultat som ger samma score.

Tabell 3: Tabell innehåller bara dem 6 högst rankade och notera att alla dem bästa modellerna innehåller Max\_depth parametern None

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rank | Max\_depth | N\_estimators | Mean fit time (s) | Mean test score |
| 1 | None | 500 | 26,64 | 0,96 |
| 2 | None | 600 | 32,67 | 0,96 |
| 3 | None | 200 | 11,17 | 0,96 |
| 4 | None | 300 | 16,49 | 0,9598 |
| 5 | None | 400 | 20,73 | 0,9598 |
| 6 | None | 100 | 5,34 | 0,9598 |

Viktigt att notera i tabellen att det knappt någon skillnad på score mellan modellerna och om modellen skulle bli vald så är det viktigt att konstatera att mer estimators kommer göra modellen onödigt större.

#### LinearSVC

LinearSVC kommer ifrån SVM och har hand om klassificeringar. Skillnaden på LinearSVC och SVC är i loss funktioner och kernel funktionen (se f4). LinearSVC är mer flexible i straff och loss funktionen och då skalar bättre i större dataset.

Eftersom i LinearSVC så är kernel tricket redan bestämt till liblinear så har denna modellen mindre hyperparametrar att välja mellan. Man har bestämst att man ska testa olika C parametrar som hanterar margin violations.

C = [0.5 , 1 , 1.5 , 2]

(Se Tabell 4) Vi kan se att ju mindre C är ju bättre är vårat resultat vilket betyder att vi tillåter flera margin violations. Det kan betyda att vi har väldigt mycket data som är svår att klassificera.

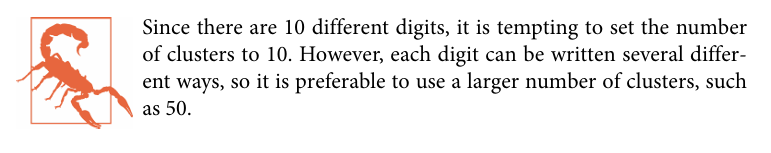
Tabell 4

|  |  |  |  |
| --- | --- | --- | --- |
| Rank | C | Mean fit time | Mean test score |
| 1 | 0.5 | 5,48 | 0,8966 |
| 2 | 1 | 6,42 | 0,8939 |
| 3 | 1.5 | 7,51 | 0,8917 |
| 4 | 2 | 7,36 | 0,8899 |

#### MiniBatchKmeans

MinibatchKmeans fungerar på samma sätt som kmeans.

Med Kmeans så är det svåra att välja kluster och det är inte lika enkelt som att använd sig utav GridsearchCV. Eftersom modellen evalueras ifrån Inertia och inte score så är det svårt för oss att jämföra denna modellen med dem andra. Detta går att lösa genom att göra lite kodning men man börjar med att försöka hitta ett bra antal kluster.

Man kan argumentera att man borde ha 10 kluster eftersom det finns 10 siffor (0-9) men man kan också argumentera att varje kluster är ett sätt att skriva en siffra och det finns fler sätt att skriva en siffra på (se Figur 8).

Figur 8: Bilden kommer ifrån boken, Hands on machine learning with scikit-learns and TensorFlow av Aurelien Geron

Med GridsearchCV kan vi mata in olika antal kluster som en hyperparameter vi har valt at testa

N\_cluster: 100, 400 700, 1000,1300,1600

Man kan se att ju fler kluster vi har ju mindre inertia får man (se Tabell 5)

Notera att Inertia är negative och det kommer från Sklearn eftersom dem anser att större är bättre och därför inverterar dem Inertian, Avstånd kan inte vara negative.

Man kan försöka leta efter en armbåga i inertia genom att kolla på minskningen i en graf vy istället (se figure 3.4)

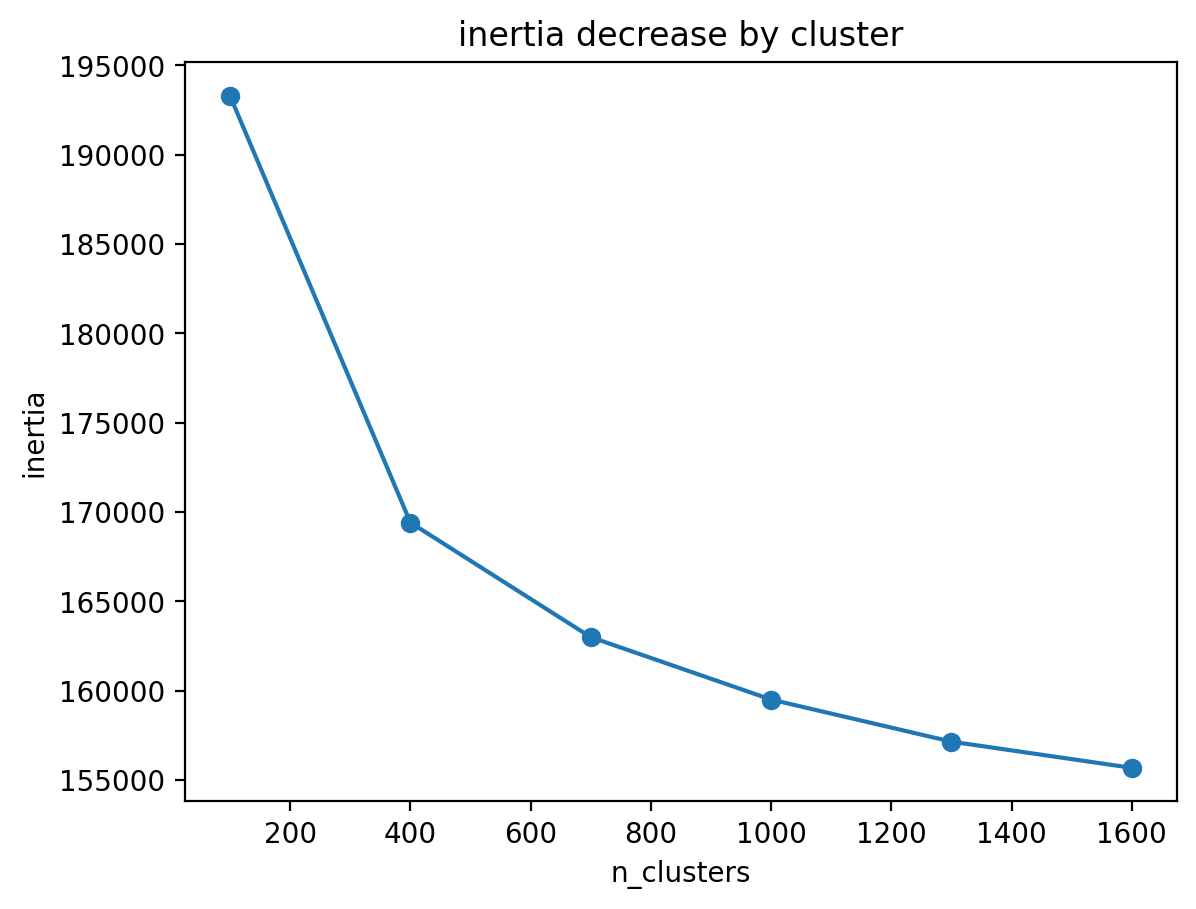
|  |  |  |  |
| --- | --- | --- | --- |
| Rank | N\_cluster | Mean fit time | Mean score (inertia) |
| 1 | 1 600 | 9,79 | -155 676 |
| 2 | 1 300 | 8,88 | -157 142 |
| 3 | 1 000 | 6,48 | -159 498 |
| 4 | 700 | 5,18 | -162 991 |
| 5 | 400 | 3,06 | -169 407 |
| 6 | 100 | 1,61 | -193 288 |

man kan se att runt 400 så finns det en armbåge men man har valt att fortsätta undersöka och väljer att kolla på förändring koefficienten mellan punkterna (se Tabell 6)

Tabell 5

Figur 9: notera att man har invertera Inertia så att den minskar istället

Figur 10



|  |  |  |  |
| --- | --- | --- | --- |
| Inertia | kluster | Första derivatan | Andra derivatan |
| 193 288 | 100 | - | - |
| 169 407 | 400 | -79,6 | - |
| 162 991 | 700 | -21,4 | 5,15 |
| 159 498 | 1 000 | -11,6 | 30,79 |
| 157 142 | 1 300 | -7,85 | 79,19 |
| 155 676 | 1 600 | -4,89 | 101 |

Tabell 6

Notera att första derivatan börja stabilisera sig runt 700-1000 cluster. Om första derivatan närmar sig 0 så kommer minskningen bli linjär och då är det menings löst att öka antalet kluster. I detta fallet säger inte andra derivatan så mycket mer än att vi är påväg ifrån en noll punkt.

Problemet här är att man inte kan jämföra denna modellen med dem andra eftersom vi inte har några labels. Det går dock och lösa om man kan lyckas sätta en label på ett kluster. Man har valt att identifiera ett kluster med antalet siffror i klustret. Genom att se vilka data punkter är i vilket kluster kan man sedan hjälp av våra labels att beräkna vilken majoritet av siffra som befinner sig i klustret. Och på så sätt kan man sätta labels på modellens kluster.

Man har dock gjort ett liknande test utan GridsearchCV eftersom GridsearchCV inte ger oss den informationen för att ge kluster en label. Man har också valt att koncentrat anta kluster närmare varandra för att som man man kan hitta en ”armbåge i armbågen”.

Detta medför dock att modellen ”predikterar” i förväg vad en siffra är för något utan någon hjälp, I en perfekt värld kommer modellen att separera alla olika siffror men i vår värld så separera dem med ca 90% accuracy (se Tabell 7: Homogeneity representerar antalet ”rena” kluster alltså att ett kluster innehåller en typ av siffra. Accuracy score är hur rätt vi har labelat alla kluster)

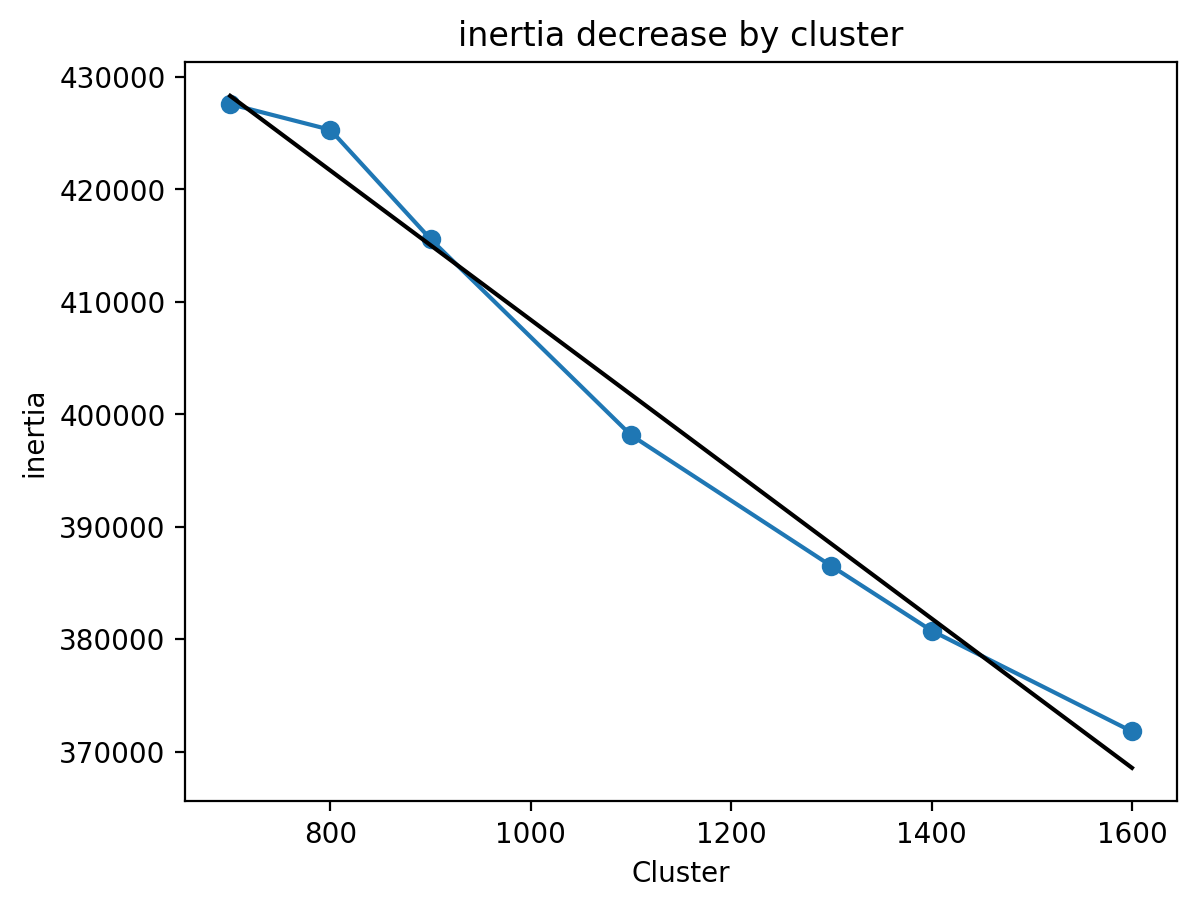
Problemet här är att medan alla andra modeller får en referens när modellen tränas så får inte kmeans det. Det är som att dem andra modellerna får testast på sin tränings data där dem fått facit och kmeans får inte det. I vilket fall som helst.

Vi kan även kolla på utveckligen i inertia i det nya testet (se Figur 11) och här ser man att inertia börjar minska linjärt mellan ökningen av cluster och minskningen av inertia. Samt att det inte framkommer någon armbåge i grafen vilket kan tolkas som att vi är förbi punkten där ökning av kluster bli menings löst.

Tabell 7: Homogeneity representerar antalet ”rena” kluster alltså att ett kluster innehåller en typ av siffra. Accuracy score är hur rätt vi har labelat alla kluster

|  |  |  |  |
| --- | --- | --- | --- |
| Kluster | Inertia | Homogeneity | Accuracy score |
| 700 | 427 573 | 0,8899 | 0,9213 |
| 800 | 425 290 | 0,8944 | 0,92225 |
| 900 | 415 571 | 0,8971 | 0,92375 |
| 1 100 | 398 136 | 0,9107 | 0,9313 |
| 1 300 | 386 497 | 0,9147 | 0,936 |
| 1 400 | 380 747 | 0,9151 | 0,93555 |
| 1 600 | 371 782 | 0,9162 | 0,9358 |

Figur 11

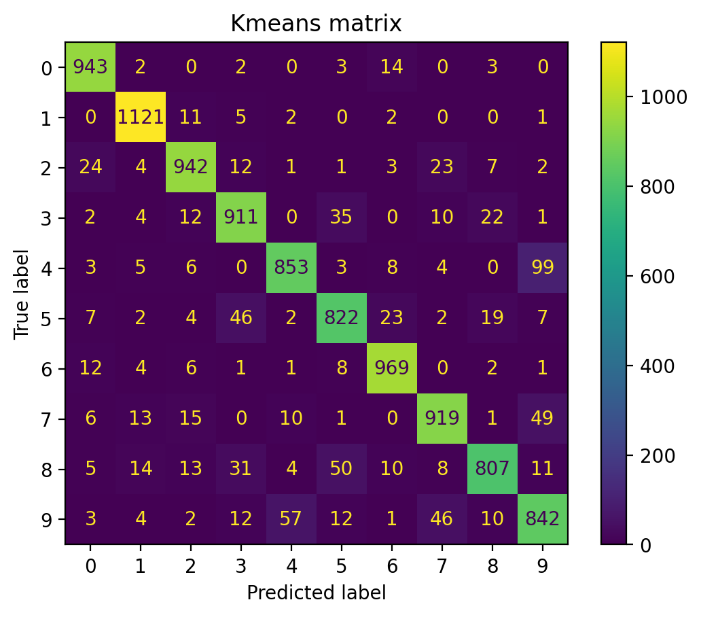
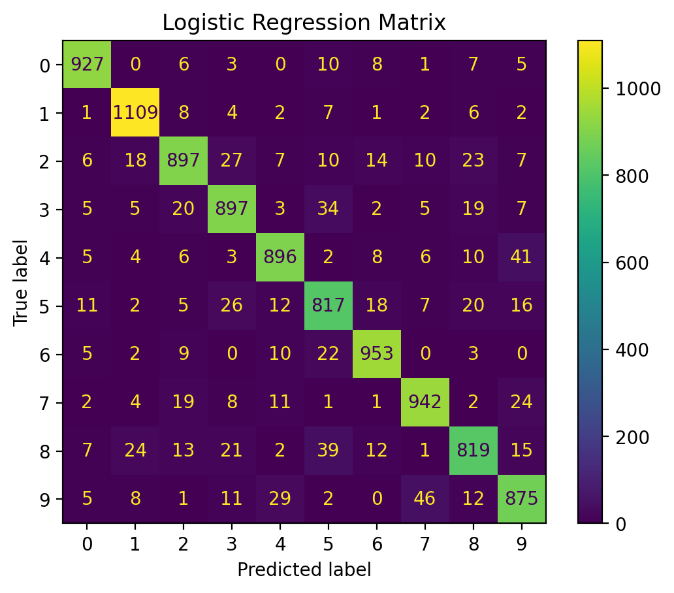


Slutligen så förekommer ett till problem med kmeans vilket är att om man har för högt antal kluster så finns de en möjlighet till att vissa kluster inte kommer innehålla någon data vilket betyder att det är omöjligt att sätta en label på det klustret. Problemat framhävs när man skall prediktera på ny data och modellen sätter in en data punkt i ett kluster som inte har en label. Detta betyder att modellen inte finner ett liknande mönster i datan och anser att den inte hör ihop med dem andra. Detta kan bero på att just den datapunkter eller siffran är skriven på ett sånt sett som modellen inte har läst förrut. I samma analys som den senaste (se Tabell 8) så kan vi se att efter 800 kluster så modellen ett få par kluster som den inte kan sätta en enda datapunkt i. Slutligen har man nästintill fullt optimerat denna modellen det finns andra vägar till att lösa dessa problemen man har valt att utgå ifrån denna analys att göra en MinibatchKmeans modell med 700 kluster

Tabell 8

|  |  |
| --- | --- |
| Kluster | Labeled Kluster |
| 700 | 700 |
| 800 | 800 |
| 900 | 899 |
| 1 100 | 1 100 |
| 1 300 | 1 299 |
| 1 400 | 1 399 |
| 1 600 | 1 598 |

#### Jämnför modell

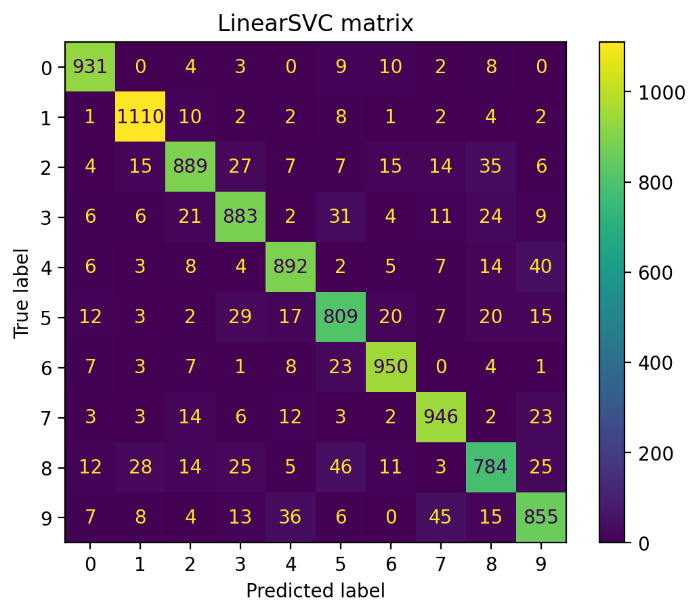
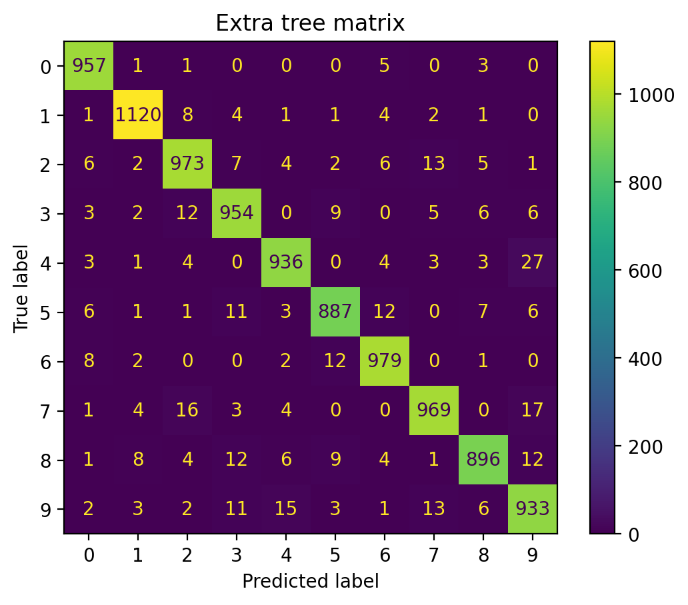
Slutligen kan Man jämföra modellerna. Man kan börja med att jämföra deras Confusion matriser (se Figur 12, Figur 13, Figur 14, Figur 15). Något att notera är att kmeans har högst fel predikteringar samt att alla modellar har en svårighet med att prediktera 5 och 8. Man kan dock se att den matrisen med bäst resultat är Extra trees. Om vi kollar på nästa tabell (se Tabell 9) så kan vi se att bästa prestanda på vårt validerings test data är Extra trees både i Accuracy score och F1 Score

Figur 12

Figur 13

Figur 14

Figur 15



Tabell 9

|  |  |  |
| --- | --- | --- |
| Modell | F1 Score | Accuracy Score |
| Extra trees | 0,9601 | 0,9604 |
| Logistic Regression | 0,9132 | 0,912 |
| MBKmeans | 0,9129 | 0,9115 |
| LinearSVC | 0,9049 | 0,9033 |

## Evaluerings process

Slutligen kan vi välja vår sista modell och träna om den på tränings och validerings datan. Man har dock valt att träna på 80% av hela MNIST data setet som inne håller 70000 bilder. Den nya tränings data kommer innehåller 56000 bilder som också kommer vara om sorterade och de resterande 14000 bilderna kommer användas som test för modellen att prediktera på.

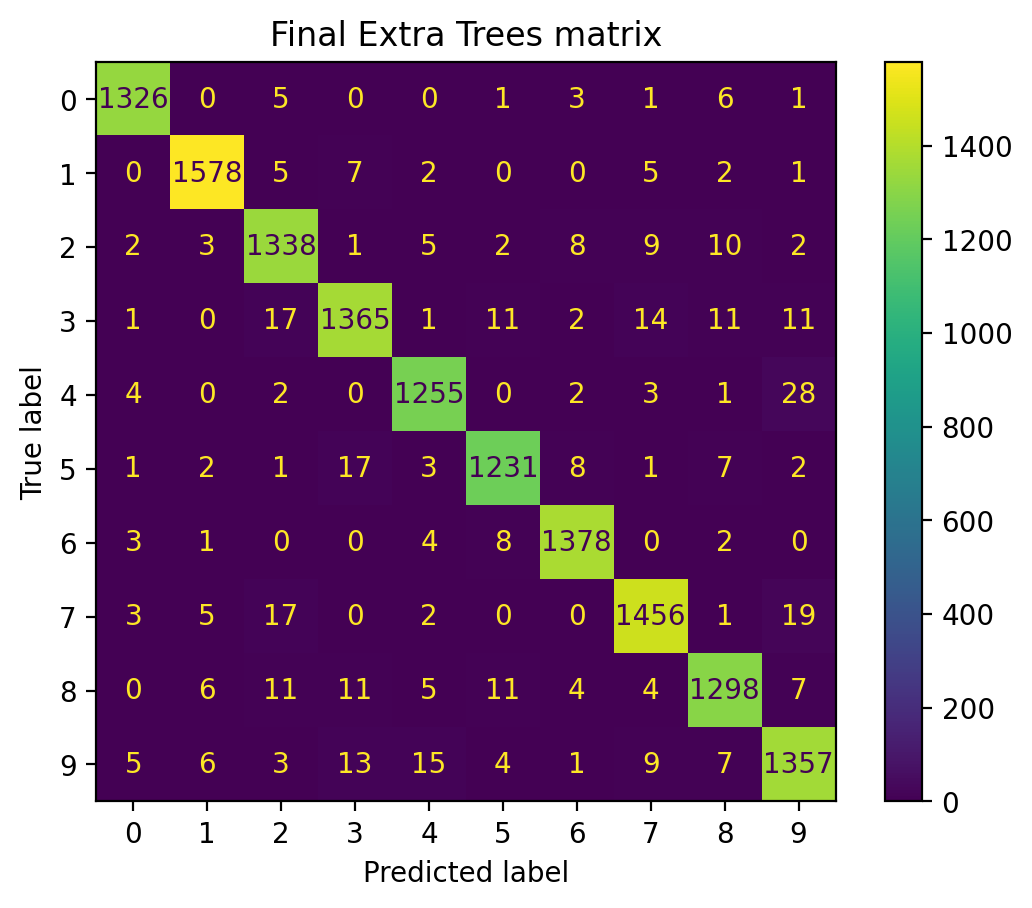
Från resultaten i Extra trees gridsearch såg man hur vissa parametrar presterade ungefär lika bra (se tabell 3.3 sida 10). Exmplevis n\_estimater = 200 prestera lika bra som n\_estimator = 500 och 600 vilket medför att vi lika gärna kan träna modellen på 200 estimators eftersom det kommer hjälpa oss tidsmässigt och utryms mässigt.

Den sista modellen presterar väldigt bra med en 97% accuracy (se Tabell 10) samt att confusion matrisen ser väldigt bra ut (se Figur 16).

Tabell 10

|  |  |
| --- | --- |
| Accuracy Score | 0,97 |
| F1 Score | 0,969 |

Figur 16



## Prediktera riktiga bilder

Modellen är tränad på handskrivna bilder så nu kan man testa modellen på egna handskrivna bilder.

Datasetet innehöll handskrivna siffror på bild som konverterat till en matris. Problemet är dock att dem bilderna är i 28x28 format vilket är en väldigt liten bild så för att modellen skall kunna prediktera en riktig bild så måste man om formatera den till ett läsbart format.

### Konvertera bilder

Med hjälp av opencv-python paketet kan man ta importerade bilder och konvertera dem till en matris med värden. Den matrisen måste sedan omvandlas till en matris med formen 28x28 och sedan skall även matrisen skalas ner som vi gjorde med datan och tillslut platta till matrisen till en dimension och matas in till våran modell.

## Webbapplikation (streamlit)

Hela process skall även göras till en webbapplikation där en användare ska själv kunna ladda upp bilder och testa modellen. Detta kan enkelt göras med Streamlit som är en plattform som är interagerat med python.

# Resultat och Diskussion

## Data

Att hantera data och omvandla data kan vara ett svårt moment. Ofta tenderar man till att använda sig av Sklearns standard scaler och vi såg en signifikant förbättring i modellens prediktioner (se Tabell 1) men varför funkar då den simpla metoden bättre?

Om man undersöker matrisen (se Tabell 11) med data och letar efter de högsta och lägsta värdena kan vi se att värdena har nästan ökat, avståndet för lägsta till högsta har ökat. Det betyda att modellen nu måste hantera större värden än förut. Men varför prestera den då bättre än den orörda datan?

Det går att argumentera att eftersom det blir ett större ”avstånd” mellan dem färgade pixlar och de icke färgade pixlarna så får modellen det enklare att klassificera siffror. Man kan spekulera om hur en normaliserad data skulle prestera på senare modeller men matematiskt sett kan man se att den normaliserade data skulle ta längre tid att träna på.

Tabell 11: Notera att värdena kommer får första iterationen av bilder

|  |  |  |
| --- | --- | --- |
| Data | Min värde | Max värde |
| Orörd data | 0 | 255 |
| Simpelt skalad data | 0,0 | 1,0 |
| Normaliserad data | -1,278 | 264,573 |

## Validering

Den största utmaningen här är att välja vilka modeller man skall ha med i valideringsprocess. Man har utgått från populära modeller som kan hantera klassificerings problem och för att få ett bredare perspektiv på hur modeller presterar så har man valt att välja modeller som skiller sig ifrån varandra. Nackdelen här är ju då att man plockar en modell som är bra på att hantera just detta problem medan de resterande modeller just hanterar dessa problem dåligt.

I ett bättre fall skulle man kunna undersöka modeller är mer kända för att hantera bildigenkännings problem, i detta fallet visade det sig att Extra trees prestera briljant på detta problem medan de andra modellerna presterade ungefär lika bra.

Det syns också i varje hyperparameters undersökning (se Tabell 2, Tabell 3, Tabell 4, Tabell 7) att det inte är någon större skillnad på ändring i hyperparametrar. Detta kan betyda att modellen är begränsad till problemet eller att man helt enkelt har testat dåliga parametrar till problemet.

### Kmeans

Något att diskutera är Kmeans kluster. Vi använde oss av flera metoder för att evaluera denna modellen då den hanteras väldigt annorlunda ifrån dem andra modellerna. Vi kan se att trots att vi valt 700 kluster så finns det andra parametrar som ger oss en bättre accuracy, men problem uppstod när kluster inte fick någon label. Det är möjligt att det kan ha funnits en lösning till detta, vi skulle kunna sätten en annan label på kluster utan någon referens och räkna dem som fel, detta kan dock medföra fler fel än rätt med våran accuracy score. Detta hade dock tillåtit en att ha flera kluster i tränings fasen.

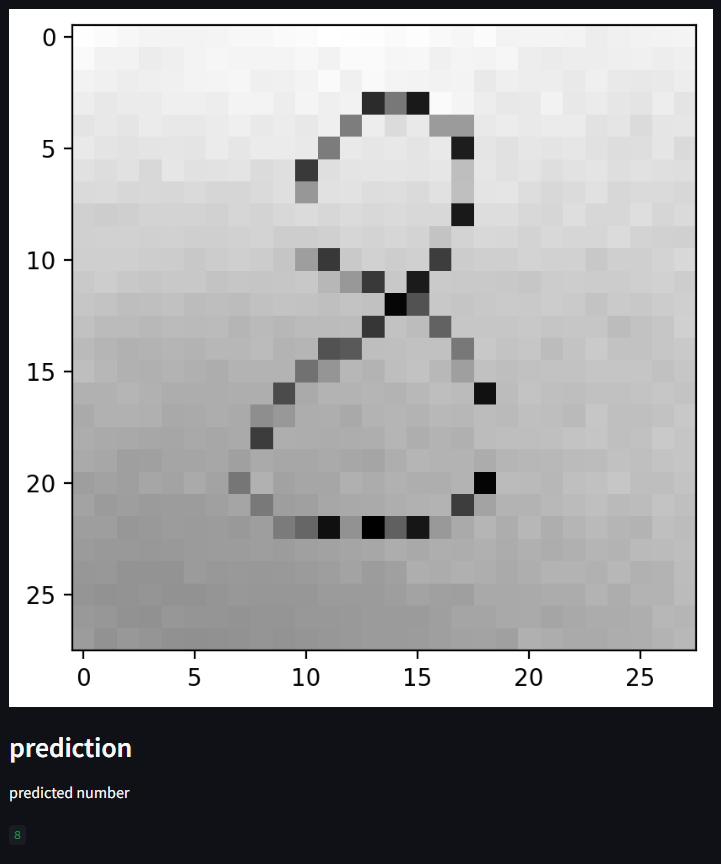
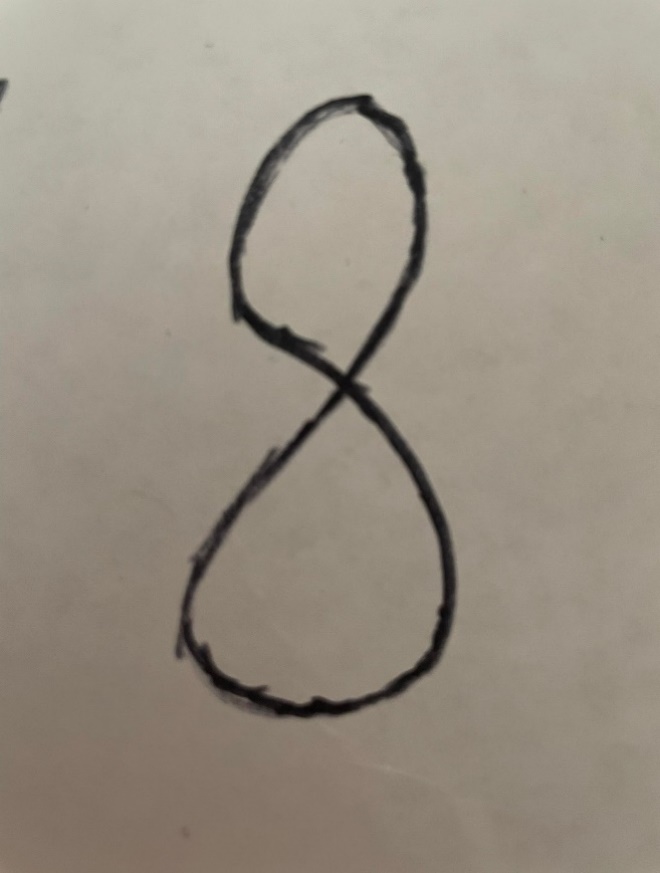
Vi kan se (se Tabell 8) att vid vissa antal kluster så får alla kluster en label och några får det inte, detta kan tolkas som att modellen har hittat X antal sätt att skriva siffror på. Ex om modellen har 900 kluster men bara 899 kluster att labela så tolkas det som att modellen bara kan hitta 899 sätt att skriva siffrorna på, Detta blir dock förvirrande efter som vid ökat antal kluster så lyckas modellen hitta fler sett att skriva siffror på eller att modellen sluta generalisera vissa sett att skriva en siffra på.

## Prediktering och Slutlig modell

Efter att ha valt modell och träna valde man att träna om modellen med den parameter som ger bäst resultat och har mins n\_estimator. Anledningen till detta är för att en Extra Trees modell tar upp väldigt mycket plats och exponentiellt ökas med antalet beslutsträd i modellen. En extra trees modell med 500 estimators blir upp till 1,5 gigabyte medan samma modell med samma data blir 200 megabyte med 200 estimators istället. Med hjälp av joblib paketet i python kan man kompremera filen ner till 81 megabyte.

Nackdelen kan dock bli att modellen försämras på ny data eller på prediktering av riktiga bilder.

Att prediktera riktiga bilder är väldigt annorlunda från att prediktera bilder från ett dataset. Vårat första problem blir att omvandla bilder till rätt dimension och format, detta medför att vi kommer bli av med vissa värden som modellen kan behöva för att göra en rätt prediktion. Kolla på exemplen (se Figur 18, Figur 17) så ser vi att modellen får vissa svårigheter när data försvinner, vi kan även diskutera vissa verkliga bilder där färgerna runt om siffrorna har kommit med som ”oljud” det är möjligt att en modell tränad med normaliserade data kan ha presterat bättre på dessa typer av bilder.



Figur 17: Handskriven 8

Figur 18: Notera, testbilder finns tillsammans med rapporten. Testa gärna modellen själv i Streamlit

# Slutsatser

Tillslut har man en modell som kan någorlunda prediktera riktiga handskrivna bilder. Våran frågeställning har blivit besvarade på ett teoretiskt sammanhang samt även hur vi använt dom metodiskt

Detta projekt är långt ifrån dagens möjligheter med bildigenkänning och maskininlärning överlag men förhoppnings vis ger detta pappret en grund för hur ML kan appliceras till både små men stora projekt. Idag ser vi AI växa i en otroligt snabb hastighet och bara ett sånt här litet projekt kan växa och utvecklas till något ännu större.

Slutligen så vill jag tacka min lärare Antonio Prgomet. Jag hänvisar även till Scikit-learns biliotek som gör det väldigt enkelt att lära sig och experimentera på egen hand.

# Teoretiska frågor

Besvara nedanstående teoretiska frågor koncist.

**1. Kalle delar upp sin data i ”Träning”, ”Validering” och ”Test”, vad används respektive del för?**

Träning används för att träna modeller, validering används för att jämföra 2 eller fler modellers resultat (att validera) och sedan används som tränings data. Test datan används i slutet i vald modell för att testa modellens ”resultat”

**2. Julia delar upp sin data i träning och test. På träningsdatan så tränar hon tre modeller; ”Linjär Regression”, ”Lasso regression” och en ”Random Forest modell”. Hur skall hon välja vilken av de tre modellerna hon skall fortsätta använda när hon inte skapat ett explicit ”validerings dataset”?**

Julia kan använda sig av krossvalidering som simpelt sett delar upp datan i mindre bitar som kallas för folds, därefter tränas modellen på flera bitar och valideras på en bit. Detta gör i flera iterationer så att olika folds får vara test och tränings data.

**3. Vad är ”regressionsproblem? Kan du ge några exempel på modeller som används och potentiella tillämpningsområden?**

Regressionsproblem är problem som hanterar kontinuerliga värden så som lön, priser och liknande. Vanliga ämnen kan va löneutveckling, försjälnings prognos, bostadsprisprognoser, Aktieutveckling osv. Modeller som används inför sånna här problem kan vara, linjär regression, Lasso regression, Ridge regression, SVR. Med mera.

**4. Hur kan du tolka RMSE och vad används det till:**

RMSE står för Root mean squared error och är ett mått för att mäta prestanda hos regressionproblem genom att beräkna skillanden från de riktiga värdet och det förutsagda från modellen. Yi-ŷi beräknar skillnaden för den punkten och dess prediktion, potensen tar bort negativa tal och 1/n tar får fram medelvärdet för alla värden. Genom att dra roten ur får vi tillbaka skalan till ett användbart värde

**5. Vad är ”klassificieringsproblem? Kan du ge några exempel på modeller som används och potentiella tillämpningsområden? Vad är en ”Confusion Matrix”?**

Klassificeringsproblem handlar om att kategorisera eller klassificera prediktioner utifrån datan. Så som att se om kunder skall lämna eller inte (churna), bildigenkänning, email spam eller inte. Modeller som används till klassificering kan vara Logistiskk regression, SVM, random forest bland annat. En Confusion Matrix är en tabell som innehåller 4 delar : True positives, False positives, True negatives och False negatives. Man analyserar dessa värden för att få insikt på modellen prestanda så som precision, recall och F1 score.

**6. Vad är K-means modellen för något? Ge ett exempel på vad det kan tillämpas på.**

k-means är en unsupervised learning modell som delar upp datan i kluster. Kan till exemple användas för kundsegmentering där med hjälp av kunders data kan dela in kunder i kluster (”grupper”) som vi sedan kan analysera för att hitta mönster eller liknande.

**7. Förklara (gärna med ett exempel): Ordinal encoding, one-hot encoding, dummy variable encoding. Se mappen ”l8” på GitHub om du behöver repetition.**

Kategorisk data skall konverteras till numeriska variabler för att användas och det kan man göra med dessa metoder. Ordinal encoding hanterar datan genom att ge den en typ av heirarki (order) t.ex mycket bra kredit = 2, bra kredit = 1, dålig kredit = 0.

One-hot och dummy variable hanterar data utan en heirarki så som färg. One-hot t.ex (blå = 1,0,0) (röd= 0,1,0) (grön=0,0,1). I dummy variable kan vi kasta en binär check och kalla 0,0 som sista t.ex (grön =0,0 )

**8. Göran påstår att datan antingen är ”ordinal” eller ”nominal”. Julia säger att detta måste tolkas. Hon ger ett exempel med att färger såsom {grön, röd, grön} generellt sett inte har någon inbördes ordning (nominal) men om du har en röd skjorta så är du vackrast på festen (ordinal) – vem har rätt?**

Julia har rätt. Det är vårat jobb som hanterar datan att klassificera den. Det är sant att en i färger finns det inte rang, men om man helt plötsligt associerar färg till något så som skönhet så uppstår det en rang.

**9. Kolla följande video om Streamlit: https://www.youtube.com/watch?v=ggDa RzPP7A&list=PLgzaMbMPEHEx9Als3F3sKKXexWnyEKH45&index=12 Och besvara följande fråga: - Vad är Streamlit för något och vad kan det användas till?**

Streamlit är ett open-source framework för att snabbt och enkelt skapa webbapplikationer för maskininlärning och dataanalys. Med hjälp av deras bibliotek kan man enkelt skapa interaktiva python-skript på webben

# Självutvärdering

1. Utmaningar du haft under arbetet samt hur du hanterat dem.

Mina största problem har varit alla problem jag stött på medan jag hanterat datan och modellerna. Speciellt att ha med en unsupervised learning modell. Jag har många gånger fått gå fram och tillbaka i kod, gjort om och gjort om mer för att få fram vissa resultat. Har stött på problem löst dem och fått fram nya problem

Att hantera fil storlekar har verkligen varit jobbigt. Vissa modeller blir förstora för github så man får träna om modeller flera gånger om vilket kan vara tidskrävande. Hade rekommenderat att visa andra joblib kompressions parameter.

En kodnings uppgift som skulle kunna ta en dag har blivit till 50 plus timmar bara på koden

Rapportmässigt. Började strukturerat men började falla ihop lite i metod delen. Hade kunnat slänga vissa meningar över till resultat och diskussions delen men de va svårt då vissa delar kopplades ihop med nästa steg i metodiken.

1. Vilket betyg du anser att du skall ha och varför.

VG, gjort en enorm analys samt fått fram en webbapplikation och en prediktions metod

1. Något du vill lyfta fram till Antonio?

Trots mina svårigheter med Unsupervised learning va det väldigt intressant, hade varit kul om det lyftes fram mer i kursen om det inte är så att det kommer mer i deep learning kursen

# Källförteckning

f1. [[1]](#endnote-1)<https://blog.exploratory.io/exploratory-weekly-update-12-3-d4b1d0f620b9>

f2. [[2]](#endnote-2)<https://datascience.stackexchange.com/questions/40900/whats-the-difference-between-sklearn-f1-score-micro-and-weighted-for-a-mult>

f3. [[3]](#endnote-3)<https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>

1. <https://blog.exploratory.io/exploratory-weekly-update-12-3-d4b1d0f620b9> [↑](#endnote-ref-1)
2. <https://datascience.stackexchange.com/questions/40900/whats-the-difference-between-sklearn-f1-score-micro-and-weighted-for-a-mult> [↑](#endnote-ref-2)
3. <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html> [↑](#endnote-ref-3)