Computer Vision

På MNIST dataset



Ian Gutierrez

EC Utbildning

Machine Learning

202403

# Abstract

The goal of this project is mainly to learn, try new thing, learn to read documentations, find information and problem solving. In this project I have utilized different models like K-Nearest Neighbour and tried different parameters to try and solve how to get an effective model to predict handwritten numbers. Overall this project show how effective machine learning can be. Have you ever tried to predict tens of thousands of numbers in under 10 minutes?

Innehåll

[Abstract 2](#_Toc162006336)

[1 Inledning 1](#_Toc162006337)

[1.1 Syfte & Frågeställning 1](#_Toc162006338)

[2 Teori 2](#_Toc162006339)

[2.1 Klassificeringsmodeller 2](#_Toc162006340)

[2.1.1 K-Nearest Neighbour 2](#_Toc162006341)

[2.1.1.1 Parametrar 2](#_Toc162006342)

[2.1.2 Voting Classifier 2](#_Toc162006343)

[2.1.2.1 Voting 2](#_Toc162006344)

[2.1.3 SVC (Support Vector Classifier) 3](#_Toc162006345)

[2.1.3.1 Regularisering 3](#_Toc162006346)

[2.1.4 Linear SVC 3](#_Toc162006347)

[2.1.4.1 Parametrar 3](#_Toc162006348)

[2.1.4.2 Regularisering 3](#_Toc162006349)

[2.1.5 Random Forest Classifier 3](#_Toc162006350)

[2.1.5.1 Parametrar 3](#_Toc162006351)

[2.2 Metriker 3](#_Toc162006352)

[2.2.1 Accuracy 3](#_Toc162006353)

[2.2.2 Confsuion Matrix 4](#_Toc162006354)

[2.3 GridSearchCV 4](#_Toc162006355)

[2.4 Streamlit 4](#_Toc162006356)

[2.5 Neurala Nätverk 5](#_Toc162006357)

[3 Metod 6](#_Toc162006358)

[3.1 Verktyg 6](#_Toc162006359)

[3.2 Datainsamling 6](#_Toc162006360)

[3.2.1 Bilder 6](#_Toc162006361)

[3.3 EDA 6](#_Toc162006362)

[3.4 Förbereda Bilder 6](#_Toc162006363)

[3.5 Träning och Testning 6](#_Toc162006364)

[3.5.1 KNN modellen 6](#_Toc162006365)

[3.5.2 Voting Classifier modellen 6](#_Toc162006366)

[3.6 Streamlit 6](#_Toc162006367)

[3.6.1 Förbereda Uppladdade Bilder 6](#_Toc162006368)

[4 Resultat och Reflektion 8](#_Toc162006369)

[4.1 Resultat 8](#_Toc162006370)

[4.2 Reflektion 8](#_Toc162006371)

[5 Slutsatser 9](#_Toc162006372)

[5.1 Testdata 9](#_Toc162006373)

[5.2 Streamlit 9](#_Toc162006374)

[6 Teoretiska frågor 10](#_Toc162006375)

[7 Självutvärdering 12](#_Toc162006376)

[8 Referenser 13](#_Toc162006377)

# Inledning

I maskininlärning är computer vision en teknologi som hjälper maskiner att känna igen bilder och uppskatta vad det är. I denna uppsatts så tittar jag närmare på MNIST dataset som är ett välkänt dataset skapat av Yann LeCun, Corinna Cortes och Christopher J.C. Burges. MNIST används ofta för träning och testning inom maskininlärning och består av 70,000 handskrivna siffror mellan 0 och 9.

I denna rapport så kommer jag att försöka få 90% träffsäkerhet på testdatan och 80% träffsäkerhet på egna handskrivna siffror. Jag kommer använda mig av modellen K-Nearest Neighbour och en Voting Classifier. Med hjälp av streamlit ska man även kunna ladda upp sina egna bilder och få de predikterade.

## Syfte & Frågeställning

Syftet med denna rapport är att försöka nå 90% träffsäkerhet på testdatan och 80% på ens egna handskrivna siffror.

Frågeställningarna jag ställer mig är: Kan jag få 90%, respektive 80% träffsäkerhet och kan jag få streamlit att fungera som tänkt.

# Teori

I denna del kommer jag förklara teorin bakom de modeller jag använt.

## Klassificeringsmodeller

En klassificeringsmodell är en modell som tittar på träningsdata eller tidigare observationer och därifrån kategorisera nya observationer som illustreras på Figur 1.

### K-Nearest Neighbour

K-Nearest Neighbour, eller KNN som den ofta kallas för, är en algoritm som använder sig av datapunkter nära sig själv för att prediktera vilken klass man tillhör. KNN går att användas för regression- och klassificeringsproblem men används oftast som en klassificerare.

(IBM, 2024)

#### Parametrar

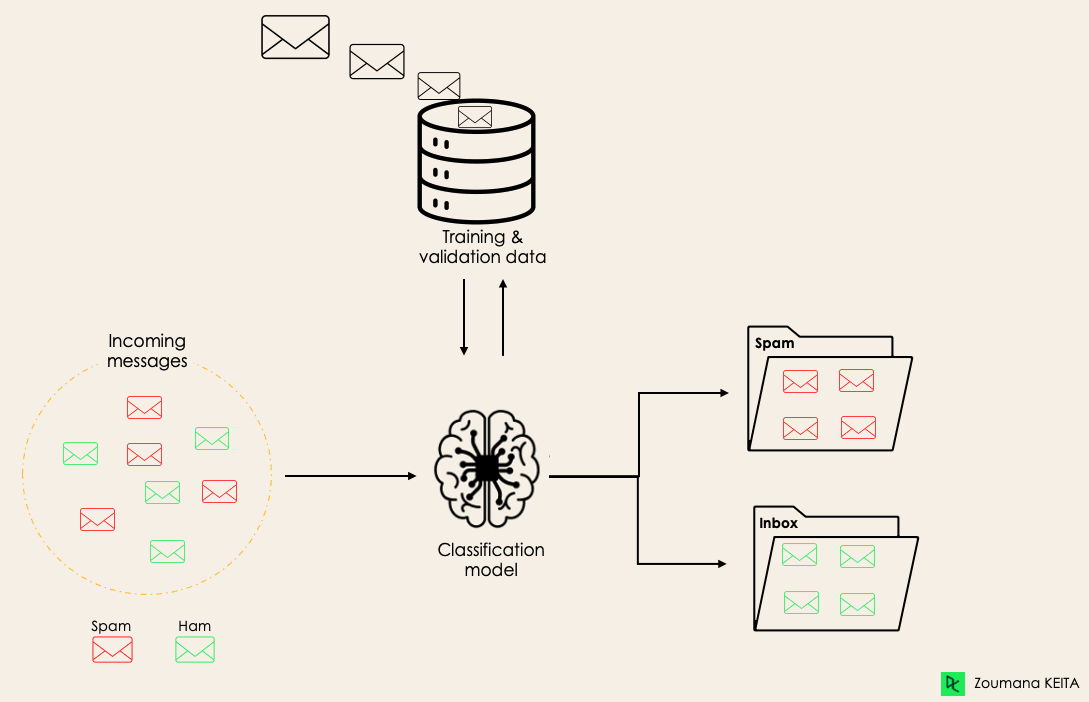
Jag kommer gå igenom de två parametrar som jag ändrade på; n\_neighbours och weights. Om vi använder n\_neighbours default värde (5) så tittar modellen på de närmaste fem ”grannarna” när den ska prediktera en ny datapunkt. Weights bestämmer hur mycket vikt varje datapunkt har, de två vanligaste valen för weights är uniform och distance, uniform lägger lika mycket vikt oavsett hur långt bort datapunkten är från det man vill prediktera. Distance gör att de punkter som är närmare ens prediktion påverkar mer. (Scikit Learn - KNeighborsClassifier, 2024)

### Voting Classifier

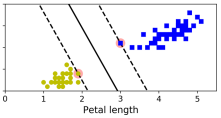
Denna klassificerare är väldigt kraftfull då den använder sig av flera olika modeller. Jag har valt att använda SVC, Random Forest Classifier och Linear SVC. Alla modeller predikterar vilken klass en datapunkt tillhör, sedan röstar alla modeller på vilket svar de tror är rätt. (Géron, 2017, ss. 189-192)

#### Voting

Det finns två olika sätt att rösta, hard eller soft. Om man använder hard voting så röstar man helt enkelt på vilken man tror är rätt. Med soft voting så tar man ett snitt för varje modell på sannolikheten för varje prediktion och sedan väljer man den modell med det högsta medelvärdet. (Scikit Learn - VotingClassifier, u.d.)



Figur 1: Klassificeringsmodell



Figur 2: SVC exempel

### SVC (Support Vector Classifier)

SVC försöker hitta det mest optimala sättet att separera olika klasser. Man brukar säga att SVC försöker skapa så bred väg som möjligt mellan två klasser, se figur 2. Skulle man med en ny datapunkt hamna innanför de streckade linjerna så kommer modellen tilldela den datapunkten en klass beroende på vilken sida av ”vägen” den är på. I de allra vanligaste fallen så måste standardisera datan för att SVC ska fungera. (Géron, 2017, s. 153)

#### Regularisering

C är SVC:s regulariseringsparameter. Den måste strikt vara positiv och använder sig av ett kvadrerad l2 (Ridge regression) straff. (Scikit Learn - SVC, 2024)

### Linear SVC

Till skillnad från SVC så behöver inte Linear SVC:s data standardiseras. Generellt så är de två modellerna väldigt lika, dock så brukar man använda Linear SVC när du har större dataset då den inte behöver standardiseras och då går snabbare. (Scikit Learn - Linear SVC, 2024)

#### Parametrar

I Linear SVC kan man själv välja vilken bestraffning man vill använda (l1 eller l2). Man kan även specificera förlustfunktionen som har standardvärdet ”hinge”. Om man använder l1 som bestraffning så kan man inte använda ”hinge”. (Scikit Learn - Linear SVC, 2024)

#### Regularisering

Precis som med SVC så kan man välja en regulariseringsparameter (C) och som med SVC så måste den vara positiv.

### Random Forest Classifier

Denna modell är baserad på beslutsträd och använder sig av slumpen för att skapa en bättre modell. Istället för att söka efter den viktigaste oberoende variabel när man delar upp en nod så söker den efter bästa oberoende variabel bland en slumpmässig delmängd av oberoende variabel. (Géron, 2017, s. 197)

#### Parametrar

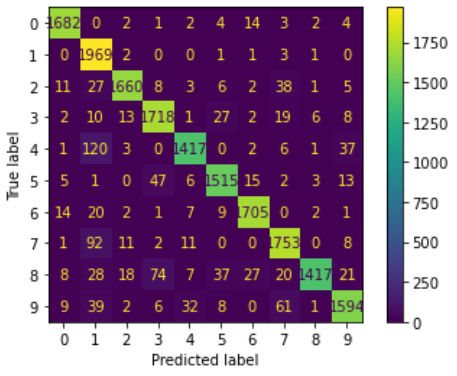
Random Forest Classifier har väldigt många parametrar men de vanligaste är n\_estimators, max\_depth, min\_samples\_split, min\_samples\_leaf och max\_features. (Scikit Learn - Random Forest Classifier, 2024)

## Metriker

Metriker är de metoder man använder för att utvärdera hur bra en klassificerare är. I mina modeller har jag använt accuracy och confusion matrix.

### Accuracy

Med denna simpla men väldigt användbara metrik så räknar man ut hur stor andel av ens prediktioner är korrekt.



Figur 3: Confusion Matrix

### Confsuion Matrix

Med en confusion matrix så kan man lätt visuellt se hur bra ens prediktioner är (se figur 3). Då vi har tio olika klasser så kommer våran confusion matrix vara 10x10 med 100 olika utkomster. På y-axeln så har vi korrekt klass, 0-9 i detta fall, och på x-axeln har vi våra prediktioner. Alltså är diagonalen där vi har predikterat rätt klass. (Géron, 2017, ss. 91-92)

## GridSearchCV

Beroende på vilken modell man använder så finns det väldigt många olika parametrar och ett extremt effektivt sätt att söka igenom om se vilka parametrar som passar bäst för just den datan och modell är GridSearchCV som testar alla parametrar man lägger in och kan ge dig en score för de olika parametrarna. (Géron, 2017, ss. 76-78)

## Streamlit

Streamlit är ett open-source bibliotek som används med python. Det används mest för att skapa applikationer för data science och maskininlärnings projekt.

En bild som visar diagram, linje, origami, mönster

Automatiskt genererad beskrivning

Figur 4: Neurala Nätverk

## Neurala Nätverk

Med neurala nätverk försöker man få maskiner att imitera biologiska neuroner genom att se mönster, överväga olika alternativ och sedan dra en slutsats. För att ett neuralt nätverk ska fungera effektivt så behöver det finnas tillräckligt med träningsdata för att få en bra träning och sedan valideringsdata för att kunna evaluera om den behöver fin justeras (se figur 4). (IBM - What is neural network?, 2024)

# Metod

I denna del kommer jag förklara hur jag fick min data, förberedandet, vilka modeller jag använde och vilka problem jag stötte på på vägen.

## Verktyg

Jag har programmerat med python, i början så använde jag Jupyter Notebook men när jag skulle börja använda streamlit så insåg jag att Jupyter inte sparar sina filer i rätt format så jag ändrade mig till Spyder.

## Datainsamling

Datan för projektet kommer från MNIST datasetet.

### Bilder

Bilderna från MNIST dataset är handskrivna siffror som är 28x28 pixlar i gråskala mellan 0 och 255 då 0 är svart och 255 är vitt.

## EDA

Som man alltid bör göra som första steg när man ska använda ett nytt dataset är det bra att läsa igenom dokumentationen om det finns någon. Sedan kollade jag om alla pixlar var antingen helt svarta (0) eller helt vita (255).

## Förbereda Bilder

Innan jag börjar träna min modell så vill jag att alla pixlar ska vara helt svarta eller helt vita så jag gjorde en simpel loop där den går igenom varje pixel och om värdet är lika med eller under 127 sätts värdet till 0 annars sätts det till 255. Sedan ville jag att alla siffror ska vara ungefär på samma plats, i mitt fall så satte jag dem längst upp i högra hörnet.

## Träning och Testning

I MNIST dataset så är ordningen blandad så att exempelvis siffrorna är i ordning vilket gör det väldigt mycket lättare att dela upp det i träningsdata, valideringsdata och testdata. Jag har använt standardvärdet (0.75/0.25) för test\_train\_split först för att dela upp i träning- och valideringsdata sedan en till split på valideringsdatan för att få validering- och testdata vilket ger mig 56.25% träningsdata, 25% testdata och 18.75% valideringsdata.

### KNN modellen

Med KNN modellen så använde jag GridSearchCV för att söka igenom flera olika parametrar och kände mig nöjd med resultatet när jag kollade på min accuracy score och confusion matrix.

### Voting Classifier modellen

Med Voting Classifier modellen så har jag använt standard värdena för varje modell (SVC, Random Forest Classifier och Linear SVC) även här så uppfyllde modellen de krav jag hade på min accuracy score och confusion matrix för att gå vidare.

## Streamlit

Som sista del så skulle jag skapa en applikation på streamlit för att kunna ladda upp bilder och få de predikterade.

### Förbereda Uppladdade Bilder

Innan prediktionen sker så har jag tre steg jag vill göra så att mina egna siffror generellt ser likadana ut som MNIST siffrorna:

* **Gråskala**: Se till att den bild jag laddar upp inte har någon färg, endast svart eller vitt.
* **Ta bort nyanser**: Använda samma loop som jag använt på MNIST siffrorna och se till att de bilder jag laddar upp endast är helt svarta eller helt vita.
* **”Centrera”**: Lägga den siffra jag laddat upp på samma ställe som jag flyttat MNIST siffrorna till.

# Resultat och Reflektion

Nu är det dags att presentera resultaten och diskutera vad man skulle kunna förbättra.

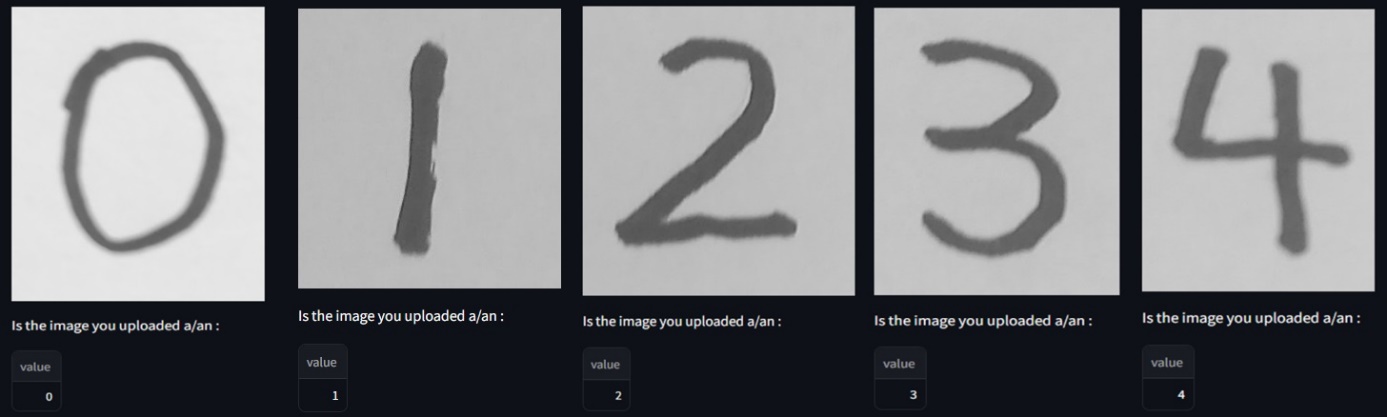
## Resultat

|  |  |  |
| --- | --- | --- |
| **Accuracy Score** | | |
| **Modeller** | **Validationdata** | **Testdata** |
| Voting Classifier | 99% | 96% |
| K-Nearest Neighbour | 98% | 94% |

Båda mina modeller uppfyllde kraven jag hade att ha minst 90% träffsäkerhet på testdatan, dock så var Voting Classifier lite bättre men väldigt mycket långsammare då den behöver träna tre modeller istället för en.

Tabell 1: Accuracy score för olika modeller

Jag valde att gå vidare med K-Nearest Neighbour modellen för att prediktera mina egna bilder då den var snabbare med snarlika resultat på testdatan. På mina egna handskrivna siffror så fick jag 90% träffsäkerhet. Då mitt mål var att få minst 80% träffsäkerhet så kände jag mig nöjd!

En bild som visar text, skärmbild, Teckensnitt, Grafik

Automatiskt genererad beskrivning

Figur 5: Resultat av prediktion på egna siffror

## Reflektion

Om jag hade gjort denna uppgift på nytt så hade jag testat flera olika modeller och testat fler parametrar, lagt mer tid på att diskutera i grupp och kollat upp hur andra hade tagit sig ann samma problem. Jag hade även satt ett högre minimum för testdatan då det var relativt lätt att få minst 90% träffsäkerhet på detta dataset.

# Slutsatser

Frågeställningarna jag ställde mig själv i början att detta projekt var:

* Kan jag få minst 90% träffsäkerhet på testdatan?
* Kan jag få minst 80% träffsäkerhet på prediktering av mina egna handskrivna siffror som jag laddar upp på streamlit?

## Testdata

Att få 90% träffsäkerhet var inte särskilt svårt att uppnå då jag med min första modell (Voting Classifier) endast behövde välja vilka modeller den skulle innehålla fick 96%.

## Streamlit

Denna del var lite knepigare då första gången jag predikterade mina bilder tyckte modellen att nio av tio siffror var sjuor. Detta tog lite grävande men efter att jag gjorde min loop som satte varje pixels värde till antingen 0 eller 255 så blev det genast mycket bättre. Efter att jag diskuterat med klasskamrater så fick jag ett tips på hur man kan centrera bilderna så att de är ungefär på samma plats, detta gjorde att jag fick 90% träffsäkerhet och uppfyllde mitt andra krav på modellen.

# Teoretiska frågor

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

Träningsdatan är till för att träna en modell.

Valideringsdata är till för att under utvecklingen justera hyperparametrar och uppskatta modellen under träning.

Testdatan är till för att evaluera modellen med data den inte tidigare har sätt.

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 ”valideringsdataset”?

Man kan använda sig av Cross-validation (Cross Model Validation).

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

Regressionsproblem är ett problem där man försöker gissa ett numeriskt värde baserat på andra oberoende variabler.

Linjär regression, Random Forest och SVM är alla bra på regressionsproblem som att förutspå personers lön eller vad ett hus i ett visst område kan kosta.

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

RMSE är medelavståndet från ens prediktioner till de sanna värdena och kan användas till att utvärdera hur bra olika modeller är.

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

Klassificieringsproblem är ett problem där man försöker gissa vilken klass man tillhör baserat på andra oberoende variabler.

Support Vecor Machines (SVM) och Logistisk regression är två modeller som kan användas för att sätta personer i en specifik klass så som om dem är unga, medel eller gamla.

En Confusion Matrix är en matris som underlättar att visuellt se hur ens modell presterar.

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

K-means är en modell som man väljer hur många kluster man vill ha, modellen skapar så många centroider och de datapunkter som är närmast en viss centroid tillhör det klustret. Exempelvis om man har information om kunders ålder och adress så kan man dela in kunder i olika kluster för att rikta olika typer av reklam beroende på vilket kluster de tillhör.

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

Alla tre är kategoriska variabler men funkar lite olika.

Ordinal encoding använder man om man tycker att det finns en tydlig ordning, exempelvis storlek på städer då Stockholm = 0, Göteborg = 1, Malmö = 2.

One-hot encoding används när det inte finns en tydlig ordning, exempelvis olika operativsystem. Man skapar lika många kategorier som det finns alternativ så om man har Windows, MacOS och Linux så skapar man tre kategorier (is\_windows, is\_macos, is\_linux) och den kategorin operativsystemet tillhör sätter man som en etta eller ”True”.

Dummy variable encoding är liknande one-hot encoding men det används alltid vid enkel linjär regression och man använder k-1 kategorier så har man samma exempel som för one-hot encoding så skapar man två kategorier, exempelvis is\_windows\_dummy och is\_macos\_dummy och skulle båda dessa kategorier vara 0 eller ”False” så använder man uteslutningsmetoden och vet att det är Linux som är rätt operativsystem.

8. Göran påstår att datan antingen är ”ordinal” eller ”nominal”. Julia säger att detta måste tolkas. Hon ger ett exempel med att färger såsom {röd, grön, blå} 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?

Jag håller med om att det behöver tolkas, man kan hitta en ordinal encoding men generellt så bör man hålla sig till nominal encoding om kanske inte alla skulle hålla med att det är ordinal encoding.

9. Kolla följande video om Streamlit: https://www.youtube.com/watch?v=ggDaRzPP7A&list=PLgzaMbMPEHEx9Als3F3sKKXexWnyEKH45&index=12 Och besvara följande fråga: - Vad är Streamlit för något och vad kan det användas till?

Streamlit är ett open-source bibliotek som används med python.

Det används mest för att skapa applikationer för data science och maskininlärnings projekt.

# Självutvärdering

1. Utmaningar du haft under arbetet samt hur du hanterat dem.   
   Största utmaningen var nog att skriva rapporten då jag aldrig skrivit en strukturerad rapport tidigare. Med hjälp av Márks och Tommys rapporter så fick jag inspiration. Att prata med andra i klassen har också hjälp väldigt mycket.
2. Vilket betyg du anser att du skall ha och varför.   
   VG – jag anser att jag uppfyller kraven för denna uppgift.
3. Något du vill lyfta fram till Antonio?

Otroligt rolig uppgift, extremt lärorik. Tyckte det var väldigt bra att du pushade om att man bör prata med andra i klassen, det har hjälp väldigt mycket att få se olika lösningar!

# Referenser

Géron, A. (2017). *Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow.* O'Reilly Media.

*IBM - What is neural network?* (den 21 03 2024). Hämtat från https://www.ibm.com/topics/neural-networks

IBM. (den 20 03 2024). *IBM*. Hämtat från https://www.ibm.com/topics/knn

*Scikit Learn - KNeighborsClassifier*. (den 21 03 2024). Hämtat från sklearn.neighbors.KNeighborsClassifier: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

*Scikit Learn - Linear SVC*. (den 21 03 2024). Hämtat från sklearn.svm.LinearSVC: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

*Scikit Learn - Random Forest Classifier*. (den 21 03 2024). Hämtat från sklearn.ensemble.RandomForestClassifier: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

*Scikit Learn - SVC*. (den 21 03 2024). Hämtat från sklearn.svm.SVC: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

*Scikit Learn - VotingClassifier*. (u.d.). Hämtat från sklearn.ensemble.VotingClassifier: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html