



**Høgskulen  
på Vestlandet**

# **DAT158 - ML Assignment 2**

## **ML Recipe Finder**

H2025

---

**Kieu-Mi Teresa Tran (676670)**

**David Lunde Sandvik (676671)**

**Andreas Tunes Huse (585911)**

# Innholdsliste

<b>Innholdsliste.....</b>	<b>1</b>
<b>1. Introduksjon.....</b>	<b>1</b>
<b>2. Datasett.....</b>	<b>1</b>
<b>3. Metode.....</b>	<b>2</b>
<b>4. Resultat.....</b>	<b>3</b>
<b>5. Web applikasjon.....</b>	<b>3</b>
<b>6. Deployment og vedlikehold.....</b>	<b>4</b>
<b>7. Diskusjon.....</b>	<b>5</b>
<b>8. Konklusjon.....</b>	<b>6</b>
<b>9. Videre arbeid.....</b>	<b>6</b>
<b>10. Vedlegg.....</b>	<b>7</b>
<b>11. Referanser.....</b>	<b>11</b>

# 1. Introduksjon

I dette prosjektet tar vi for oss hvordan maskinlæring kan brukes til å forutsi hvilken type rett en oppskrift tilhører, basert på ingrediensene den inneholder. Det er interessant fordi den kan gi en matrett og oppskrift ut fra ingrediensen en har liggende rundt. Målet er å minke mengden mat som blir kastet, grunnet at en ikke vet hvordan de kan bruke ingrediensene de allerede har hjemme. Prosjektet viser hvordan tekstdata kan omgjøres til tall og brukes i en modell som lærer sammenhengen mellom ingredienser og matrett. Det kombinerer maskinlæring i backend og moderne webteknologi i frontend.

## 2. Datasett

Datasettet som brukes i prosjektet er *Recipes Dataset : 64k Dishes* hentet fra Kaggle (Singh, 2025). Dette datasettet inneholder over 64 000 høy-kvalitet, multi-kategori oppskrifter, fordelt på mer enn 320 kategorier. Hver oppskrift har features som:

1. Tittel (*Title*), Kategori (*Category*), og Subkategori (*Subcategory*)
2. Ingredienser (*Ingredients*) (som en liste for enkel parsing)
3. Steg-for-steg Fremgangsmåte (*Step-by-step Directions*)
4. Antall ingredienser og steg (*Number of Ingredients & Steps*)

Dette datasettet er ideelt for programvarer som ønsker å anbefale oppskrifter basert på brukerens ingredienser, NLP (Natural Language Processing), og mat-relatert maskinlærings prosjekter, mm. Datasettet er forbehandlet korrekt, slik at det er ingen nullverdier i noen av featurene. Dette gjør det ideelt for gruppens prosjekt.

### 3. Metode

Siden prosjektet ikke bruker en tradisjonell tilnærming av maskinlæringsklassifisering, som Random Forest, XGBoost, Neural Networks, o.l., der en har behov for å trene datasettet, tar gruppens prosjekt heller en NLP (Natural Language Processing) tilnærming med semantisk søk. På grunn av dette, må gruppen bruke en sentence embedding-modell, som i dette tilfellet er *all-MiniLM-L6-v2* fra Hugging Face's Transformers biblioteket (Sentence Transformers, n.d.).

Dette gjør at datasettet konverteres til vektorer med fast størrelse (ved bruk av *all-MiniLM-L6-v2* modellen), som igjen gjør at vi bruker konteksten av ordene, eller ingrediensene, for å finne resultatet. Dette muliggjør også "fuzzy matching", som vil si at vi ikke trenger å skrive ingredienser nøyaktig for at konteksten skal oppfatte ordet, som for eksempel "tomatoes" og "cherry tomatoes" vil behandles noenlunde likt. Når en brukers input sammenlignes med datasettet, blir forespørselen behandlet med *cosine\_similarity* funksjonen fra *sklearn.metrics.pairwise*.

Backend er laget i Python, med forskjellige bibliotek som scikit-learn, pandas, sentence\_transformer, o.l.. For å kunne behandle forespørsler fra en klient ble det opprettet en enkel Flask API med et enkelt endpoint som kaller metoden som inneholder cosine\_similarity funksjonen.

Webapplikasjonen er bygget med React, TypeScript, og TailwindCSS. Denne sender HTTP-forespørsler til Flask APIen, som igjen gir en tilbakemelding til webapplikasjonen i JSON-format med resultatet.

## 4. Resultat

Resultatene er en embedded (vektor) fil med oppskrifter som brukes av backend når en bruker sender en request fra frontend. Samtidig har vi opprettet en moderne nettside der brukeren kan skrive inn alle ingrediensene en har tilgjengelig. Deretter får brukeren ut topp fem oppskrifter som matcher ingrediensene. Se punkt “10 Vedlegg” for skjermbilder av applikasjonen.

## 5. Web applikasjon

Den ferdige løsningen ble gjort tilgjengelig som en React webapp. Brukeren skriver inn ingrediensene de har, for eksempel “egg, milk, flour, sugar” og modellen analyserer teksten og gir forslag til hvilken type rett brukerne kan lage. Grensesnittet er enkelt og interaktivt, slik at man kan teste ulike kombinasjoner av ingredienser direkte i nettleseren. Vi valgte å bruke React for å forenkle utviklingsprosessen sammenlignet med ren HTML, spesielt fordi det gir oss tilgang til React Components og Tailwind CSS. Dette gjør at vi unngår mye av den tidskrevende feilsøkingen som kommer med vanlig CSS.

## 6. Deployment og vedlikehold

Det ble forsøkt å sette applikasjonen i drift ved å bruke Railway. Men denne deployment provideren har kun “free trial” der en kan laste opp databaser, og ikke kode. Derfor ble det bestemt at gruppen ikke setter applikasjonen i drift, men heller viser en demo. Dette blir en innspilt video der det vises hvordan embeddings genereres, og en hvordan webapplikasjonen fungerer. Demoen finnes i GitHub repository som (DEMO.mov) i root-mappen.

Dersom nye oppskrifter legges til i datasettet, er det kjapt gjort å oppdatere embeddings. Dette gjøres ved å kjøre scriptet ([start.sh](#) i “backend” mappen), som automatisk genererer embeddings. Dermed er systemet skalerbar for nye endringer, og krever lite menneskelig interaksjon for å håndtere nye oppdateringer.

Selv om systemet er relativt enkelt, kan det forbedres med å endre genereringsmodell for embeddings, dersom en ny versjon lanseres.

Webapplikasjonen er også ganske enkel, og ser moderne ut, som gjør at den ikke krever vedlikehold eller modernisering.

## 7. Diskusjon

Prosjektet demonstrerer at semantiske embeddings egner seg godt til oppskriftsøk basert på ingredienslister. Siden modellen ikke krever tradisjonell trening eller klassifisering, blir løsningen både skalerbar og enkel å oppdatere dersom datasettet endres. Likevel oppstod det flere praktiske utfordringer under utviklingen.

En av de mest merkbare problemene var variasjon i hvordan ingrediensene var representert i datasettet. Mange oppskrifter inneholdt Unicode escape-sekvenser, som `\u00bc` i stedet for faktiske tegnet  $\frac{1}{4}$ . Dette førte til at parsing-funksjonen ikke gjenkjente mengden som  $\frac{1}{4}$  cup eller  $\frac{1}{2}$  cup, og slike enheter ble vist som unicode i webapplikasjonen. Dette ble fikset med å bruke regex, som konverterer fra unicode til vanlig brøk.

Et annet problem er at modellen henter frem oppskrifter med høy semantisk likhet, men ikke nødvendigvis sikrer variasjon blant resultatene. I flere tilfeller fikk brukeren fem anbefalinger der samme oppskrift dukket opp med ulike kategorier. Dette er ikke en teknisk feil, men viser en svakhet ved å bruke rene tekstbaserte vektorberegninger uten filtrering eller logikk for duplikathåndtering. Dette er forklart mer i punkt 9. *Videre arbeid*.

En fordel ved valgt metode er at modellen håndterer “fuzzy matching”. Søk som “tomato”, “tomatoes”, og “cherry tomatoes” blir tolket som semantisk like, noe som øker treffsikkerheten og brukeropplevelsen. Samtidig betyr dette at modellen ikke nødvendigvis skiller mellom viktige nyanser, som forskjellen mellom fersk og tørket urt, eller mellom kjøtt og kjøtterstatninger.

Selv om løsningen fungerer godt, viser prosjektet at et slikt system kan driftes med relativt lav kostnad. Embeddings kan regenereres automatisk ved endringer i datasettet, og backenden krever minimalt med ressurser.

## 8. Konklusjon

Prosjektet ML Recipes har vist hvordan maskinlæring kan kombineres med moderne webteknologi for å løse et praktisk og hverdagslig problem, å finne oppskrifter basert på ingredienser man allerede har.

Ved å bruke en forhåndstrent språkmodell gjennom biblioteket *sentence-transformers*, kunne systemet representere tekst som vektorer (embedding) og dermed forstå meningen bak ord og ingredienser.

Dette gjorde det mulig å utvikle et søkesystem som finner oppskrifter som semantisk ligner på det brukeren skriver inn, for eksempel at et søk på “egg, flour, sugar” gir treff på “pancakes” og “cakes”.

Frontenden ble utviklet i React og TypeScript som gir et moderne og brukervennlig grensesnitt som lar brukeren skrive inn ingredienser, sende forespørselen til backenden og se resultatene på en ryddig måte. Backend ble bygget i Python og håndterer databehandlingen og beregner likhet mellom oppskrifter ved hjelp av cosine likhet.

Prosjektet oppnådde målet om å demonstrere hvordan språkforståelse i maskinlæringsmodeller kan brukes til å lage en intelligent oppskriftsanbefaler. Samtidig viser det hvordan slike systemer kan bygges uten å måtte trene en modell fra bunnen av ved å gjenbruke kunnskapen i ferdigtrente modeller.

Videre arbeid kan inkludere å utvide datasettet, legge til brukertilpasning (som allergier, diett eller smakspreferanser), og integrere en database for mer avanserte søk og lagring av resultater.

## 9. Videre arbeid

Som vist i vedlegg, og i demoen, kan en oppskrift tilhøre flere kategorier. Dette gjør at vi noen ganger viser topp fem resultater med samme oppskrift, men i forskjellig kategori. Forslag til videre arbeid kan derfor være å begrense resultatet, slik at vi alltid får ulike oppskrifter fra forskjellige kategorier. Dette kan for eksempel gjøres ved å sjekke tittel til oppskrift én, mot tittel til oppskrift to, og se om de er identiske. Dersom de er like, fjernes den ene oppskriften, og neste oppskrift i listen vises. Dette repeteres for alle topp fem oppskriftene.



## 10. Vedlegg

### ML Recipe Finder

Enter ingredients you have sitting around and discover recipe ideas

**Ingredients**  
Add your ingredients you have sitting around

+ Add Ingredient

Find Recipe

*Grensesnitt: Framside (uten bruker input)*

### ML Recipe Finder

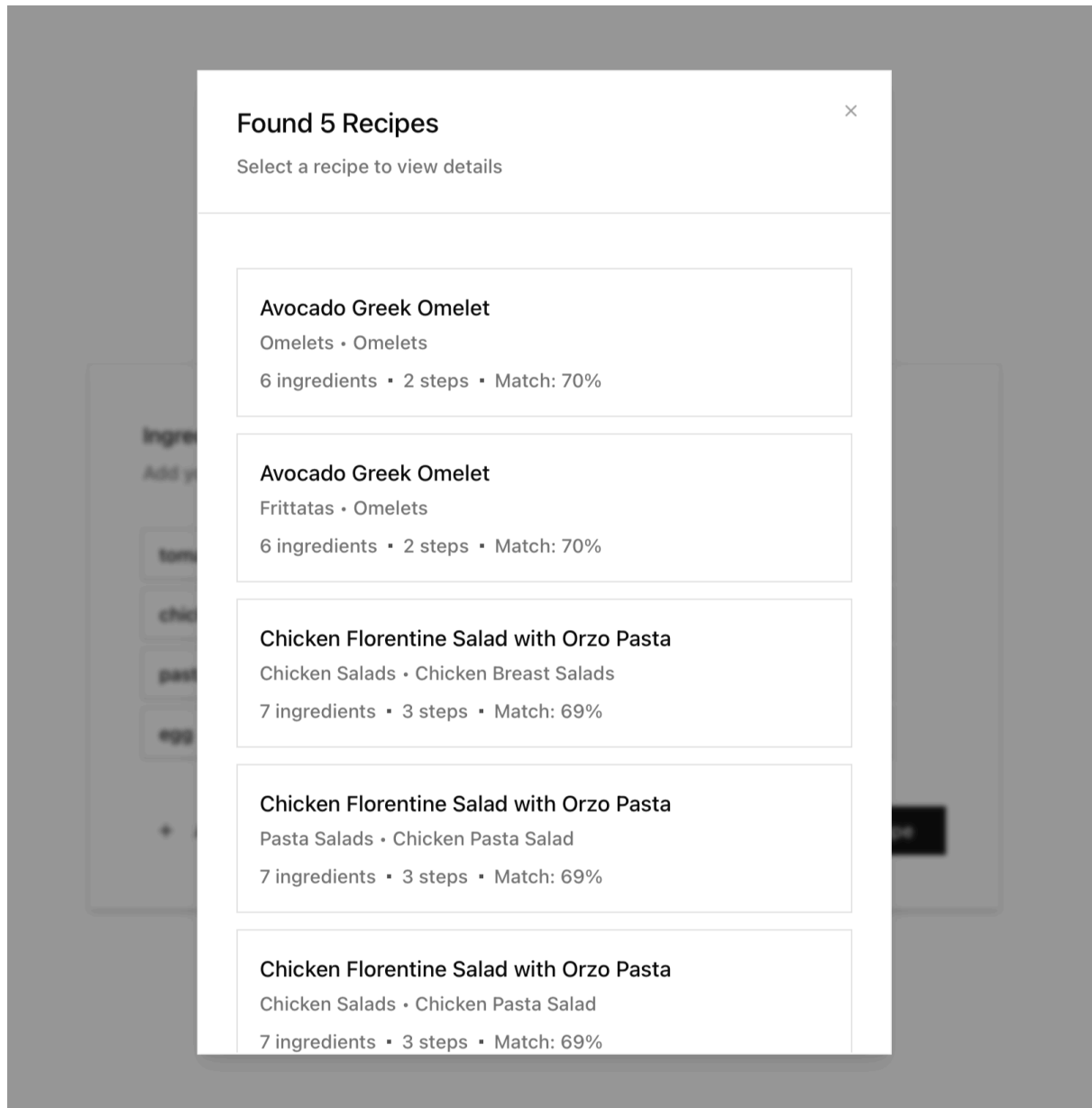
Enter ingredients you have sitting around and discover recipe ideas

**Ingredients**  
Add your ingredients you have sitting around

+ Add Ingredient

Find Recipe

*Grensesnitt: Framside (med bruker input)*



*Grensesnitt: Resultat fra backend*

## Chicken Florentine Salad with Orzo Pasta

Italian • Italian Salads

This hearty chicken Florentine salad with orzo pasta is a delicious meal filling any tummy! Serve it as a main course, and everyone will be pleased! This recipe has pleased my friends and family time and time again!

7 ingredients • 3 steps • Match: 70%

### INGREDIENTS

- 2 cups orzo pasta
- 3 cups chopped grilled chicken
- 2 (10 ounce) packages baby spinach
- 2 Roma tomatoes, diced
- 1/2 cup pine nuts
- 1/4 cup chopped sun-dried tomatoes
- 2 tablespoons capers, or to taste (Optional)

### INSTRUCTIONS

1. Bring a large pot of lightly salted water to a boil. Cook orzo in the boiling water, stirring occasionally until tender yet firm to the bite, 10 to 11 minutes. Drain and rinse under cold water.
2. Combine orzo, chicken, spinach, tomatoes, pine nuts, sun-dried tomatoes, and capers in a large bowl.
3. Whisk balsamic vinegar, olive oil, sugar, parsley, basil, garlic, oregano, salt, and black pepper together in a bowl; drizzle over salad and toss to coat. Sprinkle Parmesan cheese over salad.

*Grensesnitt: Brukeren har valgt oppskrift*

## 11. Referanser

- Sentence Transformers. (n.d.). *all-MiniLM-L6-v2*. Hugging Face. Retrieved 10. 28, 2025, from <https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2>
- Singh, P. (2025, 09 27). *Recipes Dataset : 64k Dishes*. Kaggle. Retrieved 10. 28, 2025, from <https://www.kaggle.com/datasets/prashantsingh001/recipes-dataset-64k-dishe>  
[s](#)