

# Spletna aplikacija za analizo podatkov

Končni projekt SMV

Avtor: Timotej Žavski

Mentor: Mag. Lubej Boštjann

27 February 2024

## UVOD

V svojem prostem času veliko časa nudim učenju na področju analize podatkov in izdelave umetne inteligence (LLM-Large Language Models). Moj projekt je rešitev, avtomatizacija dela, v katerem sem večkrat porabil ure in ure, vizualizacija podatkov.

Za izdelavo končnega projekta sem izbral spletno aplikacija za analizo podatkov. Popolnoma ista stvar v kombinaciji asp.net-a in pythona je moja maturitetna naloga in pri izdelavi se nisem samo naučil, kako izdelati v jeziku python spletno aplikacijo, ampak tudi kako optimizirati že obstoječo kodo. Delo na prihodnjih straneh je spremenilo strukturo in potek moje maturitetne naloge in je takšne tematike zaradi zanimanja za kariero podatkovnega analitika.

## Kazalo

Problem in rešitev	5
Obraz strani	7
Generiranje slik	9
Zaključek in povzetek	11
Viri	12

## Kazalo slik

1. Slika, Razmerje dela, (vir: privatni arhiv)	5
2. Slika, Shema aplikacije, (vir: privatni arhiv)	6
3. Slika, Uporabniški vmesnik, (vir: privatni arhiv)	7

## **Spletne aplikacije**

Spletne aplikacije so programska orodja, ki delujejo preko interneta in omogočajo uporabnikom različne funkcionalnosti, kot so komunikacija, storitve, zabava in produktivnost. Njihov pomen v današnjem času je izjemno velik, saj so postale osrednji del našega vsakdana. Omogočajo nam hitro in enostavno dostopnost do informacij, storitev ter medsebojno povezovanje, kar olajšuje in izboljšuje naše življenje tako na osebnem kot poslovnem področju.

## **Umetna inteligenca**

Umetna inteligenca (UI) predstavlja področje računalniške znanosti, ki se osredotoča na razvoj sistemov, ki lahko izvajajo naloge, ki običajno zahtevajo človeško inteligenco. ChatGPT API je ena izmed platform, ki omogoča dostop do naprednih jezikovnih modelov, ki temeljijo na tehnologiji umetne inteligence, kar omogoča avtomatizirano generiranje besedila, prevajanje, odgovarjanje na vprašanja in še več.

## **Obdelava podatkov**

Obdelava podatkov je ključna dejavnost, ki vključuje zbiranje, analizo, shranjevanje in interpretacijo informacij za pridobitev ustreznih spoznanj in podporo pri odločanju. Ta proces omogoča organizacijam, da izkoristijo vrednost svojih podatkov za optimizacijo procesov, izboljšanje produktivnosti in povečanje konkurenčnosti na trgu.

## Problem in rešitev

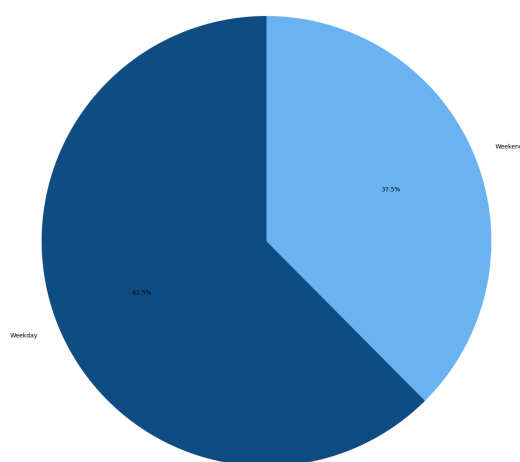
*Analiza podatkov se ne začne pri analizi.*

Analiza podatkov, kot že prej omenjeno, temelji na iskanju vzorcev v podatkih. Vzorce najlažje ljudje prepoznamo vizualno s sliko ali grafom, zato se tudi podatki vizualizirajo. Moji trditvi se lahko nasprotuje, da tudi s pregledom podatkov lahko vidimo vzorce, ampak večja količina hitro poje zmogljivosti naših možganov. To lahko tudi prikažem na primeru:

Pred sabo imamo tabelo, ki prikazuje podatke o nakupih v kavarni-pekarni obratu. Iz podatkov lahko vidimo 5 atributov in vsak ima svoj namen. Analiza je lahko tudi vizualna na podlagi podatkov: vsi izdelki so kupljeni ob skoraj isti uri na isti dan.

TransactionNo	Items	DateTime	Daypart	DayType
1	Bread	10/30/16 9:58	Morning	Weekend
2	Scandinavian	10/30/16 10:05	Morning	Weekend
2	Scandinavian	10/30/16 10:05	Morning	Weekend
3	Hot chocolate	10/30/16 10:07	Morning	Weekend

Kako bi zdaj reagirali, če vam povem, da ima ta tabela v resnici 9648 transakcij? Zdaj ni več preprosto ugotoviti, kaj je najbolj prodajan izdelek, kdaj je najbolj prodajan, katere ure so najbolj profitabilne, itd. Z vizualno analizo podatkov lahko zdaj vidimo, da potrebujemo več zaloge kave ob vikendih:

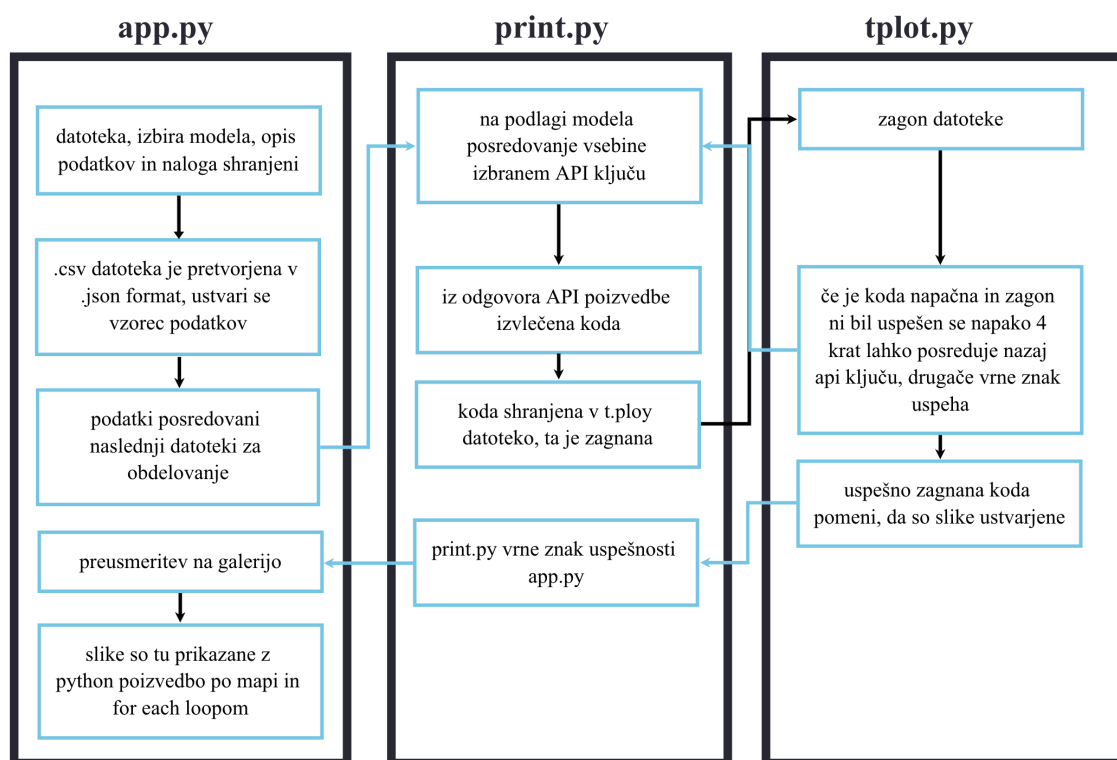


1. Slika, Razmerje dela, (vir: privatni arhiv)

Saj ima obrat v povprečju pribl. 8% vseh strank na delovni dan (temno modra) in skoraj 20% na dan v vikendu (svetlo modra).

Vizualizacija podatkov torej ogromno pripomore boljšemu razumevanju dogajanja okoli nas: ogromne mega korporacije in startupi, skupine ekosistemov in dvocelična bitja, ni važno! Vse okoli nas vključno z nami generira podatke. Problem jih ni analizirati, ampak te podatke pridobiti in jih urediti, tako da jih lahko analiziramo. Tudi, če analiza in prepoznavanje vzorcev nista zahtevna, sta zamudna in ta postopek za podjetja, ki si ne morejo privoščiti podatkovnega analitika, delno rešuje moja naloga.

Opis celotne naloge je v sledečih straneh, za pomoč pa je spodaj priložena slika, ki vodi čez korake, ki jih moje spletna stran opravi:



2. Slika, Shema aplikacije, (vir: privatni arhiv)

## Obraz strani

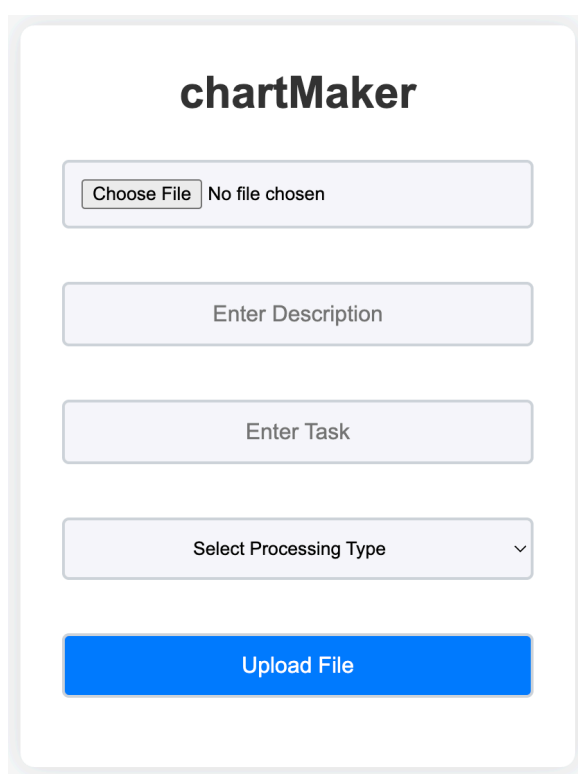
*Kaj je bilo najprej, sprednji ali zadnji del kode?*

Spletno stran sem začel z izdelavo uporabniškega vmesnika. Da lahko prikažem program kot spletno stran v Pythonu, v pomoč pride Flask. Flask je knjižnica, ki naredi ogrodje, v katerem lahko povežemo python, html, css, js. Njegov namen je prikaz spletne strani.

Ker lahko pišemo css v html datoteko tega nisem vključil, ta (ločena datoteka) je drugače v mapi static.

Struktura aplikacije v izdelovanju izgleda nekako tako:

```
moja_flask_aplikacija/  
├── static/  
│   ├── images/  
│   │   └── gen_slika.png  
├── templates/  
│   ├── index.html  
│   └── upload.html  
├── uploads/  
│   └── user?datoteka.csv  
├── requirements.txt  
├── app.py  
├── print.py  
└── tplot.py
```



The screenshot shows a web application titled "chartMaker". It features a file upload section with a "Choose File" button and the text "No file chosen". Below this is a text input field labeled "Enter Description", followed by another text input field labeled "Enter Task". There is a dropdown menu labeled "Select Processing Type" with a downward arrow. At the bottom is a prominent blue button labeled "Upload File".

3. Slika, Uporabniški vmesnik, (vir: privatni arhiv)

Slika 1. prikazuje obrazec, ki ga uporabnik izpolni. Ta vsebuje nalaganje csv datoteke, ki mora imeti urejene podatke. Opis atributov in posebnosti, ki jih lahko ima. Bolj, kot je ta opis točen, boljši je končni rezultat. Podatke, ki so naloženi moramo dobro poznati. Pred koncem uporabnik pove, kaj želi videti. Opis je lahko zelo splošen, kot "želim 5 grafov, ki mi bodo pomagali razumeti podatke bolje" ali zelo specifičen, kot "prikaži  $x$  v primerjavi z  $y$ , na prvi  $y$  osi naj bo  $y$ , na drugi  $z$ ". Preverjeno omogoča tudi 3 dimenzionalne grafe (prikaz v 2D sliki). Na koncu sledi izbira modela. Z modelom določi, kdo bo tu sprejel

vzorec podatkov, opis in nalogo. To procesiranje se lahko zgodi na računalniku lokalno, s tem podatki ne zapustijo računalnika in so za tip podatkov, ki ne smejo biti na strežniku tretjega ponudnika. Druga opcija, ki tudi v praksi boljše deluje, je pošiljanje podatkov v strežniški api. Tam jih obdela LLM GPT-4. S tem podatki zapustijo uporabnikovo napravo.

Lokalni model je model z imenom Deepseek-Coder treniran na 6.7 milijardah parametrov. Izbira tu se lahko tudi v prihodnosti poveča. GPT-4 model se odziva na naše poizvedbe z boljšimi rezultati zaradi števila parametrov, na katerih je bil model treniran, ta je 1.76 trilijona parametrov.

Prikaz slik je bil narejen kasneje, saj sem najprej potreboval delujoč zadnji del kode, ki ustvari slike. Te so prikazane v povezavi python in html. Z `app.py` najdem v mapi vse datoteke, ki se končajo na `.png` končnico. Vsa ta imena shranim v list, ki ga posredujem in kličem v `.html` datoteki. Tu imam `foreach` zanko, s katero grem čez list in prikažem vse slike z elementom `<img>`. Zaenkrat nisem našel bolj efektivne metode. V nadaljevanju bo tudi postalo jasno, zakaj ima ta smisel.



## Generiranje slik

*“Naredi vse delo namest mene!”*

Ko uporabnik pritisne gumb, se zgodi več stvari, a vzemimo eno po eno. Najprej, kar se zgodi je POST metoda, ki shrani .csv datoteko v /uploads mapo, istočasno se na UI prikaže nalagalna animacija. Stran ve, da mora naložiti novo spletno stran (upload.html), ampak čaka na odziv funkcije.

Naloženo datoteko pretvorim v .json format. V json datoteki so podatki shranjeni v blokih, s tem so lažje berljivi, razumljivi ter manj prostora za napake je. Zdaj, ko imam podatke v bolj berljivi obliki, določim par sistemskih podatkov kar pomeni, da lahko druge skripte dostopajo do njih in zaženem novo python skripto (print.py). Na tem mestu sprejemem vse nastavljene parametre. Izbiro tega procesa sem izbral, ker je v času testiranja bilo lažje odkrivanje in popravljanje napak. Takšen sistem delovanja se je obrestoval, zato sem ga tudi obdržal.

V print.py najprej uvozim potrebne knjižnice, takšna, ki izstopa in jo je predčasno potrebno naložiti je uradna knjižnica OpenAI, ki poskrbi za komunikacijo z chatGPT API ključem. Vključil sem tudi requirements.txt, ki ima potrebne knjižnice za lažji prvi zagon uporabniku.

S posebno funkcijo naprej kličem conf.py, ki ima moj api ključ. Ta ne sme biti shranjen na Githubu ali znotraj kode. Pred komunikacijo z api, definiram nalogo, ki jo je dal uporabnik in kontekst. Tega določim sam in je v pomoč pri oblikovanju zelenega odgovora. V mojem primeru ima specifična navodila, kako mora izgledati koda ki jo vrne, da vedno vrne kodo, kako jo vrne, itd. Tu je bistvo programa in na temu ‘fine-tunu’ temelji kvaliteta grafov in posledično aplikacije.

Spomnimo se, kodo jo API vrne, shranim v tplot.py datoteko, ki jo zaženemo - ta generira slike. Zato dam v kontekst kjer so navodila tudi kje je json datoteka, vzorec podatkov (vzorec vsebuje prva 2 bloka podatkov iz json datoteke) in opis podatkov. Opisano zgoraj je rešitev na problem neekonomskega pošiljanja celotne datoteke skozi api ključ.

V svojem primeru tu uporabljam GPT-4 verzijo z oknom razumevanja 128 tisoč žetonov. Odgovor, ki ga dobim ni pod nobenim pogojem takoj uporabljen, iz njega je treba izluščiti kodo, jo shraniti v novo datoteko in to zagnati. V primeru, da zagon datoteke javi napako, napako vzamem in jo z istim kontekstom kot prej pošljem nazaj preko API z navodilom naj popravi obstoječo kodo. Imam implementirano tudi funkcijo, ki v primeru da je prvič generirana koda ustvarila n število grafov, ampak ne vseh, prebere vsa imena slik in te v kontekstu posreduje nazaj z napako, da GPT ve, za katere slike ne potrebuje ponovno generirati kode. S to metodo prihranimo tudi na zapravljenih žetonih, ki niso zastonj.

Ko se zanka, ki ima maksimalno 4 poskusov (omejitev da preprečim neskončno komunikacijo med programom in api-ključem) konča, so vsi podatki iz /upload mape izbrisati, prav tako je očiščena tplot.py datoteka in tako pripravljena za prihodnjo uporabo. Takšno čiščenje uporabnikovih 'smeti' služi dobro pri iskanju napak in število teh zmanjša, vsaj po mojih izkušnjah.

Na tej točki z JS preusmerim na ustvarjen link. To ponovno naredim z generiranjem znakov v python programu in posredovanju teh v JS odsek kode, ki je v index.html. Ker je kup znakov variable tega lahko uporabim kot link.

Slike so zgenerirane, sledi njihov prikaz. Kot prej omenjeno, jaz ne vem in ne morem nadzorovati imena slik. Zato prikažem vse slike, ki so v mapi /images. To naredim z pregledom mape in hranjenjem vseh imen, ki imejo .png končnico v list. Vse to, v povezavi z zanko uporabim za dinamičen prikaz vseh slik.

Za prikaz slik uporabljam tudi dve knjižnici, ki sem ju našel na internetu. Tako je uporabnikova izkušnja prijaznejša in dinamična.

## **Zaključek in povzetek**

Moje delo je osebni projekt, ki se je začel v popolnoma drugačni obliki več kot polovico leto nazaj. Pri delu sem se naučil potrpežljivosti pri reševanju problemov in iskanju napak. Pridobil sem tudi mehke veščine testiranja, saj je to časovno vzelo velik del mojega časa.

Pri delu sem velikokrat tudi prišel do problema, kjer moj zelo zmogljiv računalnik ni bil kos delu, ki sem ga opravljal. Rezultat tega je večkrat bila optimizacija kode, ki kot večšina vedno pride prav, občasno pa tudi izposoja zmogljivejše opreme v oblaku za namene testiranja in pogona težjih operacij. Poleg tega je strošek tudi bil uporaba API storitve podjetja OpenAI, do datuma pisanja je taga za slabih 20eur.

## Viri

- deepseek-coder (<https://ollama.com/library/deepseek-coder>, 10. 2. 2024).
- ollama-python GitHub repository (<https://github.com/ollama/ollama-python>, 5. 2. 2004)
- Ollama GitHub repository (<https://github.com/ollama/ollama>, 2. 2. 2004)
- Flask documentation (<https://flask.palletsprojects.com/en/3.0.x/>, 26. 1. 2024)
- How To Display Images In Python Flask (<https://plainenglish.io/blog/how-to-display-images-in-python-flask>, 24. 2. 2024)
- OpenAI API key documentation for Python (<https://platform.openai.com/docs/api-reference>, 10. 8. 2023)