Prediktív elemzés végzése egészségügyi adatokon Machine Learning modellek segítségével

Perform predictive analytics on health data using Machine Learning models

Viczián Dániel
ZV7QI5

Tóth László, Senior mérnöktanár

2024

Neumann János Egyetem

Neumann János Egyetem
GAMF Műszaki és Informatikai Kar, Informatika Tanszék

Szakdolgozati feladatlap

Név: Viczián Dániel **Neptun kód:** ZV7QI5
Szak: Mérnökinformatikus

Specializáció: mobil és web fejlesztési **Munkarend:** levelező

A szakdolgozat címe

Prediktív elemzés végzése egészségügyi adatokon Machine Learning modellek segítségével

Kibocsájtó tanszék: Informatika Tanszék
Kutató Csoport: INFO - Szoftvertechnológiai Szakcsoport
Belső konzulens neve: Tóth László
Beosztása: oktató

A feladat részletezése:

1. Adatok gyűjtése, előkészítés, tisztítás és adatbázis tervezés
2. Használt Machine Learning modellek bemutatása, összehasonlítása, értékelése
3. Predikció szívkoszorúér-betegség kockázatának előrejelzésére
4. Alvási rendellenességek felismerése osztályzási feladattal
5. Az elkészült alkalmazás tesztelése

Kelt.: Kecskemét, 2024. október 25.

Tóth László s.k.
oktató

Dr. Pásztor Attila s.k.
Tanszékvezető

Tartalomjegyzék

Bevezetés

A szakdolgozat célja egy olyan alkalmazás fejlesztése, amely az adattudomány eszközeit és módszereit felhasználva képes hatékony adatfeldolgozásra, elemzésre és predikcióra. Az adattudomány több szakterületet kombinál, így a dolgozat készítése során lehetőség nyílik különböző tudományágak mélyebb megismerésére, valamint gyakorlati alkalmazására. Ezek a tudományágak közé tartozik például az adatelemzés, a statisztika, a valószínűségszámítás, a gépi tanulás, az adatfeldolgozás, valamint az informatika egyéb ágazatai. Az elkészített webes alkalmazás az adatelemzési eredményekre épül, amelyekből gépi tanulási modellek segítségével különböző előrejelzéseket készíthetünk.

A fejlesztés során a Python programozási nyelvet használom, amely az adattudomány egyik legalapvetőbb eszköze. Ennek egyik legfőbb oka, hogy a Python rendelkezik egy kiterjedt, széles körben támogatott könyvtárkészlettel, amely kifejezetten ennek a területnek a feladataihoz készült. Ezek a könyvtárak eszközöket biztosítanak a gépi tanulási algoritmusok használatához, az adatok elemzéséhez és feldolgozásához, valamint az eredmények vizualizálásához.

Egy felügyelt gépi tanulási modell alapja a tanító adatbázis. Számos terület adatbázisai közül lehetett választani. Ilyen terület például az egészségügy, gazdaság, sporttudomány, marketing. Végül egészségügyi adatokra esett a választás, mivel így saját, valós adataimmal is lehet tesztelni az elkészült predikciós modelleket egy interaktív felületen. Ehhez szükség volt egy API kapcsolatra, amin keresztül okosóráról érkező adatokat lehet fogadni, majd pedig eltárolni azokat egy saját adatbázisban. A kiválasztott tanító adatbázisok két különböző egészségügyi kockázatelemzést végzez el: szívkoszorúér-betegség kockázatának előrejelzésére, valamint az alvási rendellenességek felismerésére. A kapott eredményeket a gépi tanulási modellek különböző paramétereinek finomhangolása előzi meg, hogy a lehető legpontosabb előrejelzést biztosítsák.

Az alkalmazás a Streamlit webes keretrendszer segítségével valósul meg. A felület lehetőséget biztosít adatvizualizációk megjelenítésére, valamint a gépi tanulási modellek interaktív bemutatására. A végső verzióban az előzetesen tesztelt és legjobban teljesítő gépi tanulási modellt előrejelzéseinek az eredményei jelenek meg a felületen.

1. Projekt bemutatása

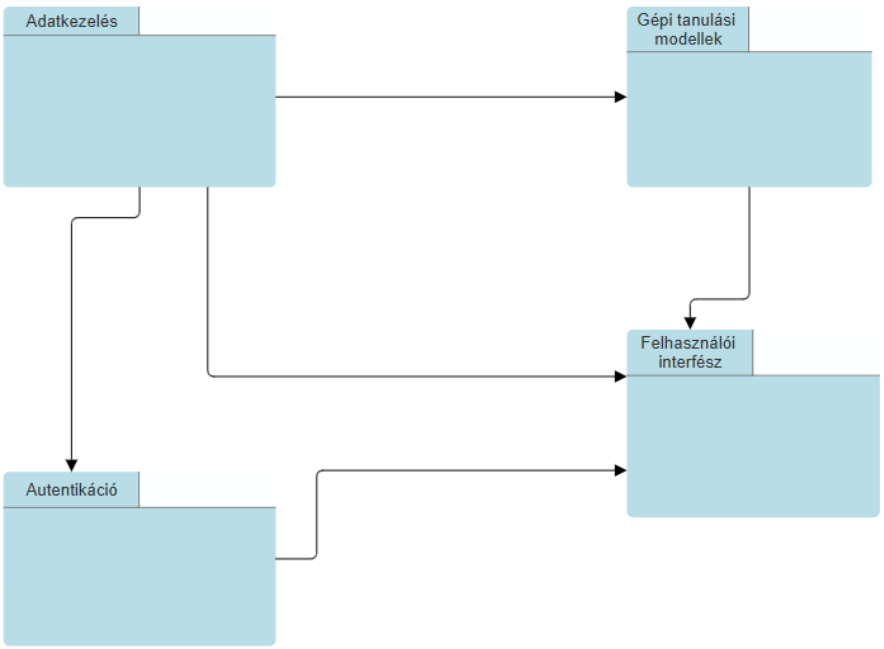
1.1.Projekt felépítése

A projekt fejlesztése a Visual Studio Code fejlesztői környezetben történt. A használt programozási nyelv a Python, mivel támogat számos könyvtárat adattudomány területről, például a gépi tanulási feladatokhoz és az adatvizualizációhoz. A Python emellett felhasználóbarát programozási nyelv, mivel egyszerű és könnyen olvasható a szintaxisa, amely lehetővé teszi, hogy viszonylag gyorsan megírjuk a kódot.

Az alkalmazás belépési pontja az app.py fájl, amely kezeli az alkalmazás navigációját és tartalmazza a futtatáshoz szükséges kódot. Ezen felül 4 fő könyvtár alkotja a projektet:

- auth (Autentikáció)
- data (Adatkezelés)
- main (Felhasználói interfész)
- models (Gépi tanulási Modellek)

Az auth könyvtárban található az auth.py fájl, amely kezeli az autentikációt. A data csomagban szerepel az összes adatbázisműveletet tartalmazó modul. A main az alkalmazás felhasználói felületeit tartalmazza, a model pedig a két predikcióhoz szükséges metódusokat. A könyvtárstruktúrát és a köztük lévő kapcsolatot az alábbi csomagdiagram szemlélteti.



1 ábra: Könyvtárstruktúra csomagdiagramja

1.2.Használt könyvtárak és keretrendszerek ismertetése

Az alkalmazás készítése során jó néhány Python könyvtár telepítésére volt szükség. A különböző csomagok telepítési és a használatához szükséges információkat a PyPI weboldal biztosította. Ez egy nyilvános gyűjtemény Python könyvtárakból és modulokból, amely lehetővé teszi a fejlesztők számára, hogy különféle előre megírt kódokat, funkciókat és eszközöket könnyen beépítsenek a saját projektjeikbe. [1]

1.2.1. Streamlit

A Streamlit egy nyílt forráskódú Python keretrendszer, amely lehetővé teszi adatelemző és gépi tanulási webalkalmazások gyors és egyszerű létrehozását. A Streamlit segítségével egyszerűen átalakítható a Python szkript egy jól működő és könnyen használható webes alkalmazássá. Emellett a gépi tanulási modellek eredményei interaktívan jeleníthetők meg, és dinamikus diagramok készítésére is alkalmas. A Streamlit nem támogatja az URL-routing-ot, azaz a különálló oldalak kezelését. Az oldalak közötti navigációt azonban lehet szimulálni például egy oldalsáv (sidebar) segítségével, ahol a navigálás során dinamikusan frissül a tartalom.

1.2.2. Scikit-learn

A Scikit-learn egy nyílt forráskódú Python könyvtár, amelyet gépi tanulási algoritmusokhoz használnak. Ez az egyik legnépszerűbb eszköz az adatelemzés és gépi tanulás területén, mivel egyszerű és jól dokumentált segédletet biztosít különféle feladatokhoz. A Scikit-learn alapja a NumPy, a SciPy és a Matplotlib, amely biztosítja a numerikus számításokat, az adatelemzést és a vizualizációt. A könyvtár különböző osztályozó, regressziós és klaszterező algoritmusokat tartalmaz, beleértve a Support Vector Machine-t, Random Forest-et, K-Nearest Neighbor-t és a logisztikus regressziót. A gépi tanulási modellek algoritmusai mellett a Scikit-learn-t használtam a modellek teljesítményének értékelésére, valamint az adattisztítási és adatelemző műveletekhez is.[2]

1.2.3. Pandas

A Pandas egy Python könyvtár, ami adatelemzésre és adatkezelésre alkalmas. Hatékonyan lehet adatot beolvasni különféle forrásokból, mint például CSV, Excel vagy SQL adatbázisokból. Mivel a tanító adatbázisok CSV formátumúak a programban, Pandas segítségével lettek beolvasva a nyers adatok. A könyvtár emellett eszközöket biztosít az adatfeldolgozásra, mint például az adatok tisztítására és átalakítására. A programban szintén a Pandas felelt az SQL-lekérdezések eredményeinek beolvasásáért és egy DataFrame-be töltéséért, valamint a felhasználók által a formon keresztül beküldött adatok feldolgozásáért.

1.2.4. NumPy

A NumPy (Numerical Python) egy nyílt forráskódú Python könyvtár, amely alapvető eszköz az adatelemzésben, különösen a numerikus számításokhoz és tömbök kezeléséhez. Ez az egyik legtöbbet használt könyvtár ezen a területen, mivel gyors és hatékony tömbműveleteket tesz lehetővé, és sok más adatfeldolgozó könyvtár, mint például a Pandas, és a Scikit-learn is erre épül. A Pandas háttérben NumPy tömböket használ az adatok tárolására és kezelésére. Az alkalmazásban a NumPy-t elsősorban numerikus tömbök kezelésére használom, amelyek az adatvizualizáció során lényegesek.

1.2.5. Plotly

A Plotly egy nyílt forráskódú interaktív grafikus könyvtár, amely használható különféle típusú ábrák készítésére és megjelenítésére Pythonban, JavaScript-ben, R-ben, és más nyelvekben is. A legfőbb előnyei, hogy lehetővé teszi az interaktív vizualizációt és számos különféle ábrát támogat: például vonaldiagramokat, hisztogramokat, dobozdiagramokat, hőtérképeket, földrajzi térképeket, 3D ábrákat. A Plotly könnyen integrálható különféle adatelemző könyvtárakkal, mint a Pandas, NumPy. Az alkalmazásban az első predikció eredménye egy Plotly által megjelenített gauge (mérőóra) diagrammal van szemléltetve. A Garmin Connect API segítségével lekért adatokat szintén ennek a könyvtárnak segítségével jelennek meg egy külön felületen interaktív vonaldiagram formájában.

1.2.6. Matplotlib

A Matplotlib egy széles körben használt grafikus Python könyvtár, amely adatvizualizációra szolgál. Elsősorban statikus ábrák megjelenítése a feladata. Különösen hasznos, ha gyors és egyszerű adatvizualizációra van szükség. A Plotly-hoz hasonlóan a Matplotlib is kitűnően integrálható más elemzési könyvtárakkal, mint például a Pandas és a NumPy. Az alkalmazásban az adattisztítás és elemzés, illetve a modellek összehasonlításakor megjelenített diagramok ábrázolására használtam a Matplotlib-et, mivel ezeknél nem volt szükség összetettebb, dinamikus megjelenítésre, amelyre a Plotly alkalmasabb lett volna.

1.2.7. Seaborn

A Seaborn egy szintén adatvizualizációs könyvtár, amely a Matplotlib alapjaira épül, és egyszerűbben lehet a segítségével különféle ábrákat megjeleníteni. Különösen jól használható statisztikai vizualizációkra, és kiegészíti a Matplotlib-et azáltal, hogy alapértelmezés szerint szebb és rendezettebb grafikonokat készít, kevesebb kódolással. A programban ezt a könyvtárat használtam például az elemzések során a korrelációs mátrix és a kiugró értékeket megjelenítő boxplot diagram készítésére.

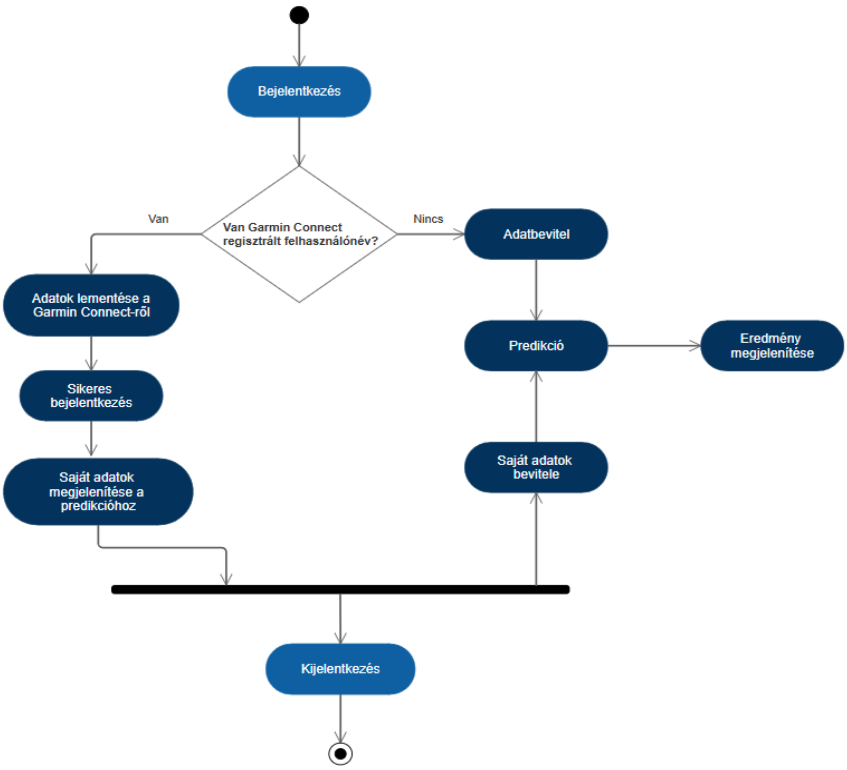
1.2.8. PostgreSQL

A PostgreSQL egy fejlett objektum-relációs adatbázis-kezelő rendszer, amelyet széles körben használnak megbízható, nagy méretű adatbázisok kezelésére. A PostgreSQL támogatja az SQL szabványt, és számos funkcióval rendelkezik, mint például tranzakciókezelés, összetett lekérdezések, adatkonzisztencia és a skálázhatóság. A PostgreSQL jól használható eszköz adattudomány feladatok elvégzésére, mivel nagy méretű adathalmazt lehet tárolni a segítségével.

Az adatbázis és a Python szkript közötti kapcsolatot az SQLAlchemy és psycopg2 könyvtár biztosítja. Az SQLAlchemy egy magasabb szintű eszköz, amely megkönnyíti az adatbázisokkal való munkát, és lehetőséget nyújt az ORM (Object-Relational Mapping) használatára. Az psycopg2 pedig a PostgreSQL adatbázis alacsonyabb szintű meghajtója, amely közvetlen kapcsolatot biztosít az adatbázissal. [3]

1.3.Bejelentkezési folyamat

A predikciókat tartalmazó oldalak bejelentkezés nélkül is elérhetőek, azonban a bejelentkezés lehetőséget biztosít a felhasználónak arra, hogy saját valós adatait használja. A bejelentkezés opciója az oldalsávban található. A bejelentkezéshez a Garmin Connect alkalmazásban regisztrált felhasználónév és jelszó szükséges. A bejelentkezési folyamatot egy tevékenységdiagrammal szemléltethető.



2 ábra: Bejelentkezési folyamat tevékenységdiagramja

A helyes bejelentkezési adatok megadása után a rendszer API segítségével csatlakozik a Garmin Connect alkalmazáshoz, ahonnan lekéri a szükséges adatokat, majd eltárolja azokat az adatbázisban. Ezt követően az oldalsávban megjelennek a lekért adatokból számított értékek, amelyek segítségével pontos és valódi értékekkel lehet paraméterekkel ellátni a predikciós modelleket. Az oldalsávban megjelenő érték az adott változó elemeiből számított medián. A lekért adatok egy külön felületen is ábrázolva vannak, amelyek sikeres bejelentkezés után az oldalsávban megjelenő Garmin Connect oldalon érhetőek el. Ezeket az értékeket idősoros vonaldiagramon tekintheti meg a bejelentkezett felhasználó.

Amennyiben érzékeny adatokat, például jelszavakat tárolunk, abban az esetben gondoskodni kell azok megfelelő titkosításról. A titkosításra többféle csomag áll rendelkezésre. Kezdetben a Bcrypt könyvtár titkosító algoritmusát alkalmaztam, azonban amikor az elkészült alkalmazás feltöltésre került a felhőbe, problémák léptek fel a függőségek kezelésével kapcsolatban, aminek a Bcrypt volt az egyik oka. Ezért alternatív megoldást kerestem a titkosításra.

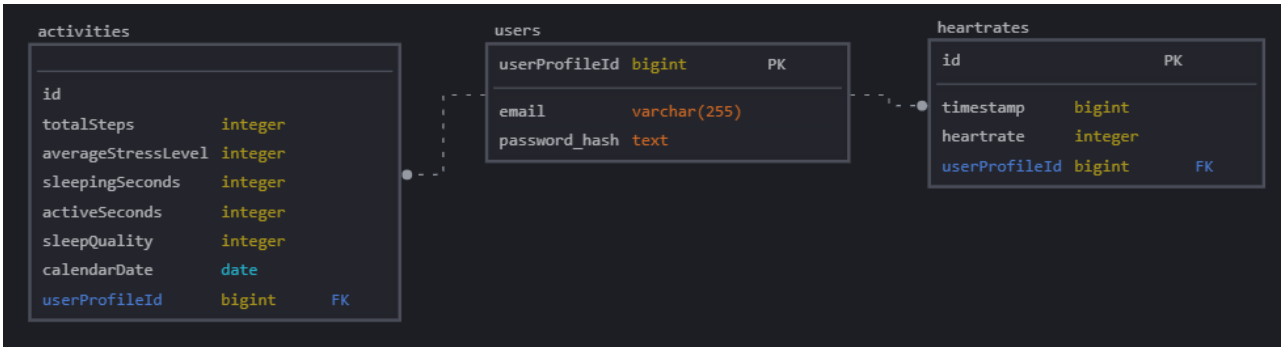
A titkosítást végül a hashlib beépített Python könyvtár segítségével történik. Egy függvény gondoskodik arról, hogy az eredeti jelszavakat ne közvetlenül tároljuk, ami fontos biztonsági szempont. Az os.urandom() függvény segítségével egy véletlenszerű 16 byte méretű salt-ot generálunk, amely minden egyes hash-elési folyamat során eltérő, ami azt jelenti, hogy még azonos jelszavak esetén is különböző hasheket kapunk. Ezzel az eljárással tovább növelhető a biztonság.

```
def hash_password(password):
    salt = os.urandom(16)
    hashed_password = hashlib.pbkdf2_hmac('sha256', password.encode('utf-8'), salt, 100000)
    return salt + hashed_password
```

1.3.1. Adatbázis

A lekért adatok a Garmin API-n keresztül JSON formátumban érkeznek. Az adatok feldolgozása során a JSON állomány tartalmát átalakítjuk, majd az így előkészített adatokat elmentjük a PostgreSQL adatbázisba. Ez a folyamat biztosítja, hogy az adatok megfelelő struktúrában kerüljenek tárolásra, és később könnyen hozzáférhetők legyenek elemzés vagy vizualizáció céljából. Az adatok három különböző táblába tároljuk:

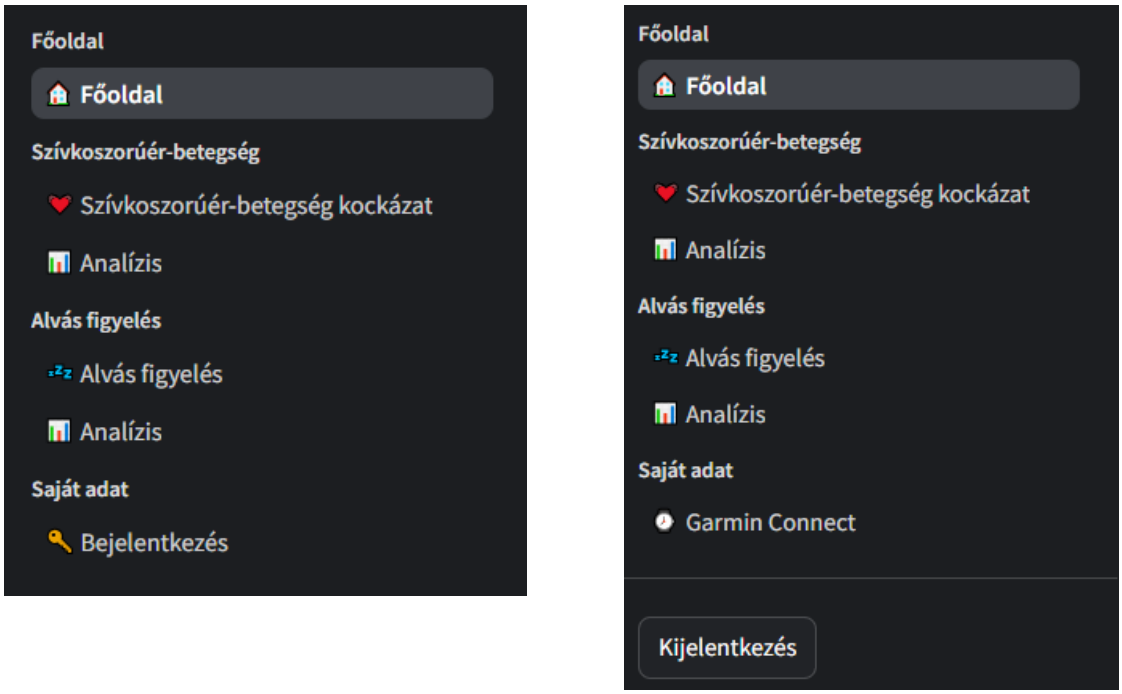
- activities: Az alvási rendellenesség felimerésére szolgáló adatokat tartalmazza.
- heartrates: A két feladathoz megjelenített pulzus értékhez szükséges adatok.
- users: A felhasználó bejelentkezési adatait tartalmazó tábla.



3. ábra: Adatbázis felépítése

1.4.Az alkalmazás funkciói

Az alkalmazás futtatásakor a főoldal tölt be elsőnek, ahol rövid ismertető található az oldalról. A Streamlit különböző funkciókat biztosító felületei között navigálni egy oldalsáv (sidebar) segítségével lehet. Az alkalmazás autentikációs funkcióval rendelkezik, ezért a navigációs felület eltérő lehetőségeket biztosít attól függően, hogy a felhasználó be van-e jelentkezve.



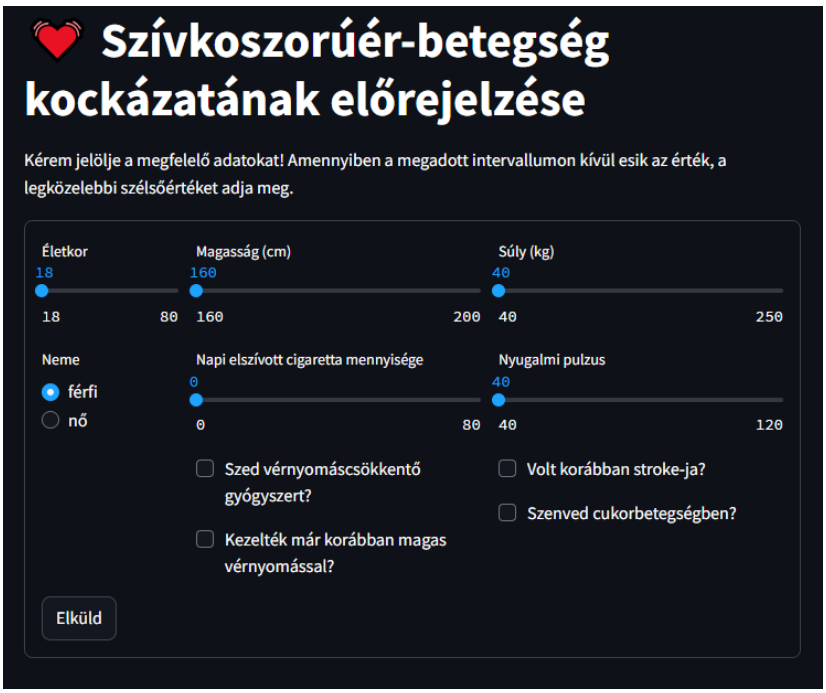
a ***b***

4. ábra: A navigációs felület

a – Bejelentkezés előtt; b – Bejelentkezés után

A predikciókat el lehet végezni bejelentkezés nélkül is. Mindkettő feladathoz tartozik egy oldal, ahol egy űrlapon keresztül van lehetőség kitölteni az adatokat. A numerikus adatokat egy csúszka (slider) segítségével lehet megadni. A két szélsőértéket a tanító adatbázis alapján

határoztam meg, mivel az intervallumon kívül eső értékek torzíthatják a modell eredményeit, ezért volt szükség az intervallum rögzítésére.



a

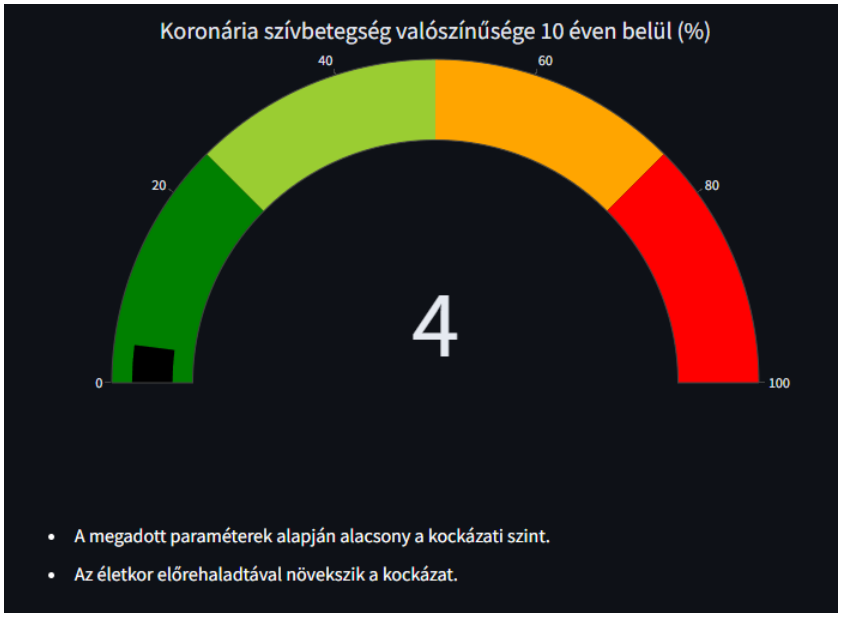
b

5. ábra: Az űrlapok

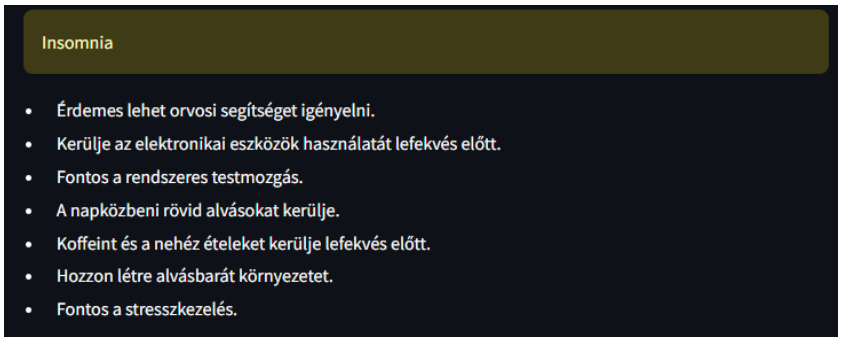
a – Szívkoszorúér feladat; b – Alvási rendellenesség feladat

Az űrlap kitöltése után a megadott adatok alapján kiszámítja a modell az eredményt. Emellett tartozik mindkét feladathoz egy Analízis oldal, ahol a szakdolgozathoz használt diagramok találhatóak. Például az adattisztítási műveletekhez szükséges ábrák, illetve a modellek teljesítményének összehasonlításához használt diagramok.

A szívkoszorúér-betegség kockázati feladat valószínűség százalékos eredménye egy gauge diagramon van ábrázolva. Az alvás figyelés esetében pedig a becsült osztály szöveges értéket adja vissza. Az eredmények mellett megjelenik néhány egészségügyi ajánlás mindkét feladatnál, amely változik a végső érték függvényében.



a



b

6. ábra: A predikciók eredményei

a – Szívkoszorúér feladat; b – Alvási rendellenesség feladat

2. Adattisztítás és előkészítés

A felügyelt Machine Learning modellek által végzett megbízható becslés alapfeltétele egy pontos és kellően reprezentatív tanító adatbázis. Az alkalmazáshoz használt tanító adatbázisok a Kaggle weboldalról származnak, ahol számos adatkészlet áll rendelkezésre specifikusan adatelemző és gépi tanulás feladatok elvégzésére. A nyers adatok CSV formátumban vannak eltárolva, ezeken elvégzem az adattisztítási műveleteket, majd az elkészült végleges adattáblák a PostgreSQL adatbázisba kerülnek feltöltésre. A modellek teljesítménye nagyban függenek a végső adatbázis minőségétől, ezért indokolt a különböző változók mélyebb megismerése és bemutatása. [4]

A két előrejelzésére szolgáló kezdeti adatbázis változóit az alábbi táblázatok mutatják be.

1. táblázat: Tanító adatbázis a szívkoszorúér-betegség kockázathoz

Változó	Leírás	Típus
male	*páciens neme (férfi 1, nő 0)*	*bináris*
age	*páciens kora*	*numerikus*
education	*végzettség*	*numerikus*
currentSmoker	*dohányzik-e*	*bináris*
cigsPerDay	*napi elszívott cigaretta mennyiség*	*numerikus*
BPMeds	*Vérnyomás csökkentő gyógyszert szed-e a páciens?*	*bináris*
prevalentStroke	*Volt-e a páciensnek korábban stroke-ja?*	*bináris*
prevalentHyp	*Szenvedett-e a páciens magas vérnyomásban?*	*bináris*
diabetes	*Van-e diabétesze a páciensnek?*	*bináris*
totChol	*Koleszterinszint*	*numerikus*
sysBP	*a szívverés alatti vérnyomás*	*float*
diaBP	*szívverések közötti vérnyomás*	*float*
BMI	*páciens testtömegindexe*	*float*
heartRate	*páciens pulzusa*	*numerikus*
glucose	*páciens glukózszintje*	*numerikus*
TenYearCHD	*Volt-e 10 éven belül szívkoszorúér-betegség? (függő változó)*	*bináris*

2. táblázat: Tanító adatbázis az alvási rendellenességek felismeréséhez

Változó	Leírás	Típus
Gender	*páciens neme*	*string*
Age	*páciens kora*	*numerikus*
Occupation	*foglalkozás*	*string*
Sleep Duration	*alvás hossza (h)*	*float*
Quality of Sleep	*alvás minősége (1-10)*	*numerikus*
Physical Activity Level	*aktivitás szint (min/nap)*	*numerikus*
Stress Level	*stressz-szint (1-10)*	*numerikus*
BMI Category	*páciens testtömegindexe*	*string*
Blood Pressure	*vérnyomás (systolic/diastolic)*	*string*
Heart Rate	*pulzus*	*numerikus*
Daily Steps	*napi lépésszám*	*numerikus*
Sleep Disorder	*alvási rendellenesség (None, Insomnia, Sleep Apnea), függő változó*	*string*

2.1.Adattisztítás a kardiovaszkuláris problémákat előrejelzésére szolgáló tanító adatbázisban

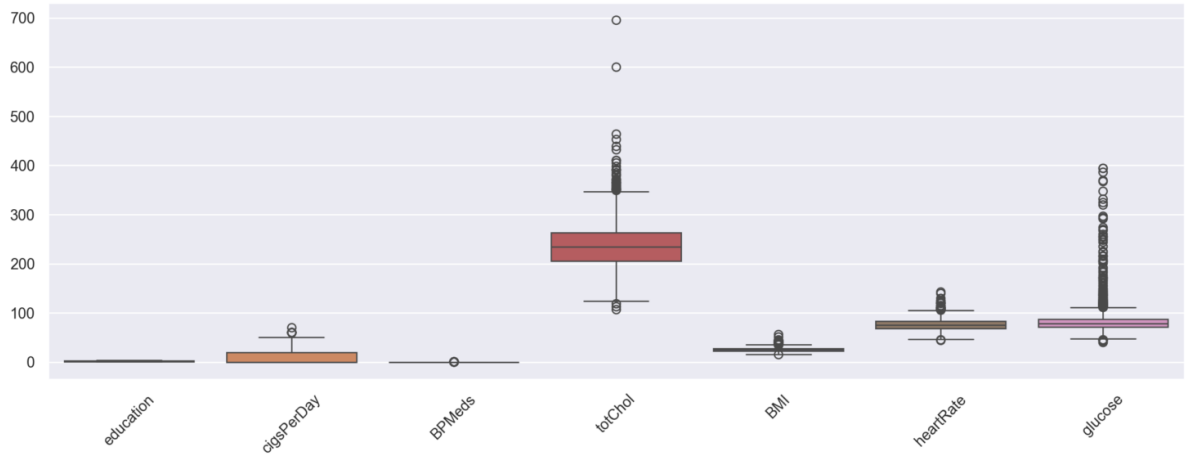
2.1.1. Hiányzó adatok kezelése

Az első lépés a hiányzó adatok azonosítása az adatbázisban. Amennyiben található ilyen mező akkor pótolni kell, vagy egyszerűen elhagyhatóak a hiányzó adatokat tartalmazó sorok. A vizsgált adatbázis 4238 rekordot tartalmaz, amelyből 645 cellában hiányzik adat, ami az összes cella körülbelül 1%-át jelenti. Általános ajánlások szerint, ha a hiányos sorok teljesen véletlenszerűen oszlanak el, illetve a hiányzó sorok aránya nem haladja meg az 5%-ot, abban az esetben el lehet hagyni a megfelelő sorokat. Ezúttal mégsem az adatok elhagyását választottam, hanem a pótlást. Mivel így nagyobb eséllyel kapunk pontosabb eredményt az előrejelzésre. [5]

Különböző módszerek állnak rendelkezésre hiányzó adatok pótlására, például átlaggal, mediánnal vagy regressziós alapú becsléssel történő pótlás. Amennyiben az adataink bináris értékek, vagy szöveges értékkel rendelkeznek, abban esetén alkalmazható a módusszal való pótlás. Tehát meg kell vizsgálni a kiválasztott oszlopok típusát. A vizsgálatot a Pandas Python könyvtárának a nunique() függvényét használom. Aminek a segítségével megállapítható, hogy hány egyedi érték található az adott oszlopban.

Amennyiben az adataink bináris értékek, vagy kategorikusak, abban esetén alkalmazható a módusszal való pótlás. A BPmeds nevű oszlop kettő egyedi értéket tartalmaz, ami 1 és 0 lehet, tehát bináris. Az education nevű oszlop kategorikus, míg többi oszlop numerikus adatokat tartalmaz.

Ha az adataink numerikusak akkor a medián érték használata azért lehet előnyös, mert kevésbé érzékeny a szélsőséges, kiugró értékekre. A kiugró értékeket vizuálisan is ellenőrzöm box plot diagram segítségével, amely jól szemlélteti az egyes oszlopok adatainak eloszlását és a kiugró értékeket.

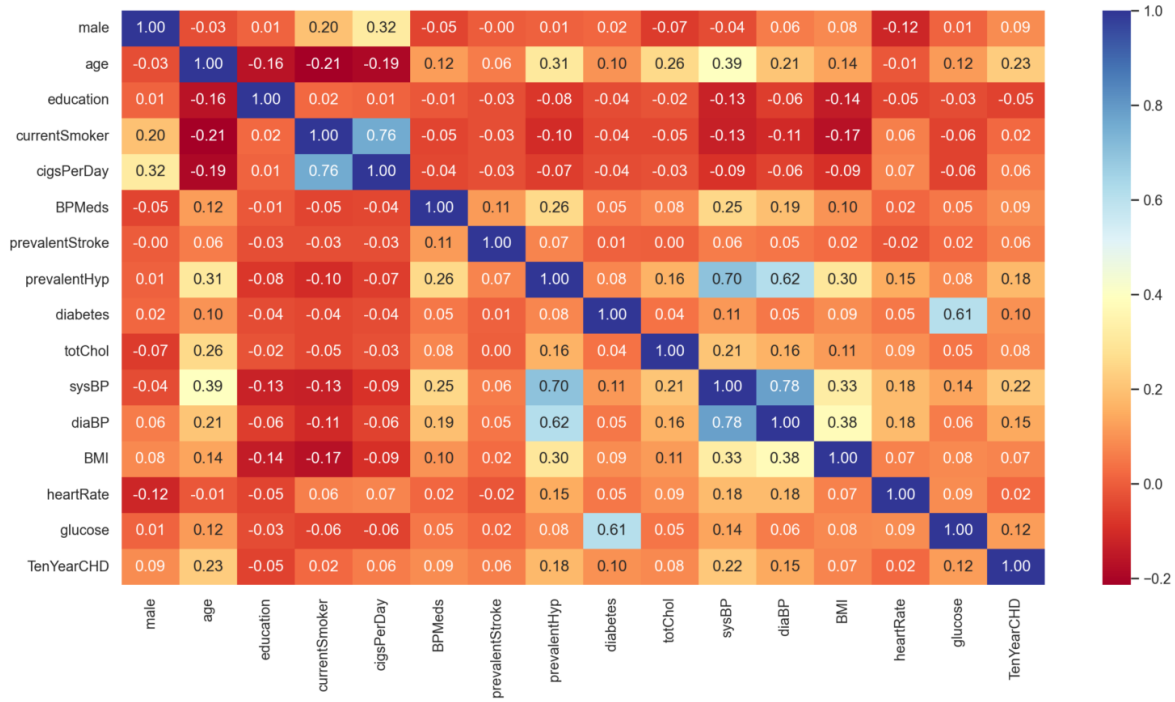


7. ábra: Kiugró értékek

A box plot diagram elemzése után leolvasható, hogy sok kiugró érték található, ezért az adatok mediánnal való pótlása megfelelő választásnak tűnik, mivel ez a módszer minimalizálja a kiugró értékek torzító hatását.

2.1.2. Korreláció vizsgálat

Fontos megvizsgálni az összes független változó korrelációját a függő változóval, valamint ellenőrizni a multikollinearitás meglétét. A multikollinearitás azt jelenti, hogy a független változók között túl szoros a korrelációs kapcsolat, amely negatívan befolyásolhatja a becslést. A korrelációk vizsgálatára egy hatékony módszer a korrelációs mátrix, amely leírja az összes változó egymás közti kapcsolatát. [6]



8. ábra: A szívkoszorúér-betegség kockázat adatainak korrelációs mátrixa

A korrelációs mátrix vizsgálata után könnyedén megállapíthatóak a változók közötti kapcsolat erőssége. A korrelációs együtthatók megmutatják, hogy az egyes változók között milyen erős a kapcsolat. Az értékek -1 és +1 között mozognak:

- +1: Tökéletes pozitív korreláció – ha az egyik változó növekszik, a másik is mindig növekszik.
- 0: Nincs korreláció – a két változó között nincs egyértelmű lineáris kapcsolat.
- -1: Tökéletes negatív korreláció – ha az egyik változó növekszik, a másik csökken.

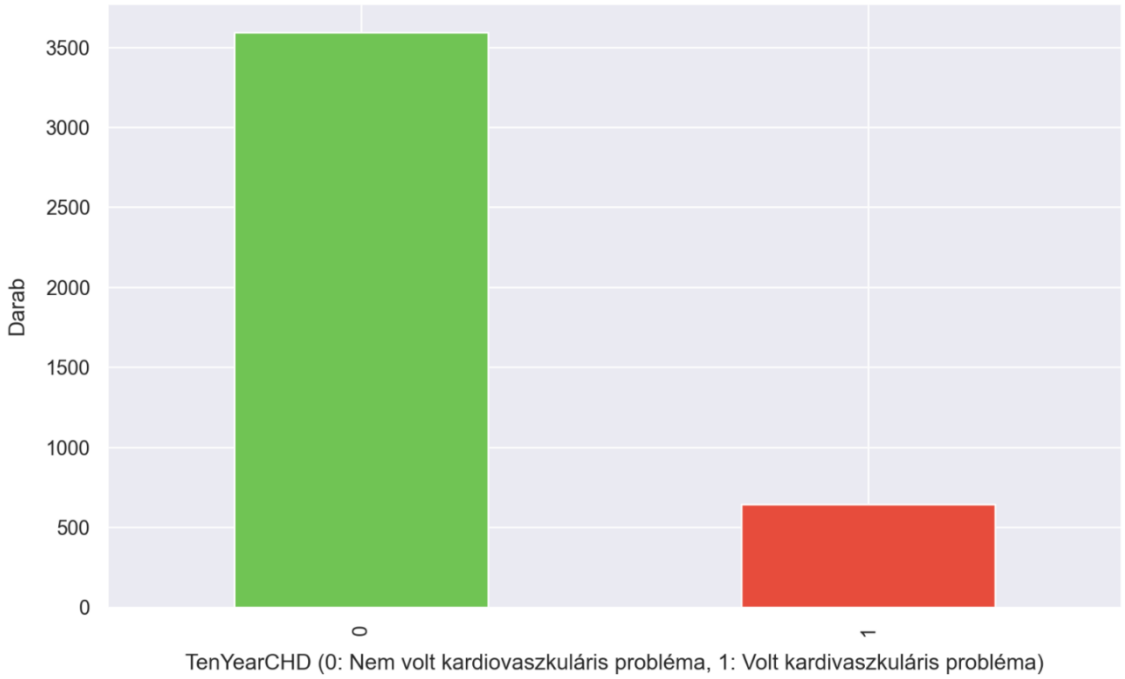
Ahol két változó között magas a korreláció, elhagyom az egyik változót, mivel a multikorreláció ronthatja a modell teljesítményét, különösen lináris regresszió esetében. Például 0,7 feletti a korreláció mértéke a currentSmoker és a cigsPerDay változók között. Ami logikus is mivel a currentSmoker változó egy bináris érték arra, hogy dohányzik-e a páciens. A cigsPerDay változó pedig a napi elszívott cigaretta mennyisége. Itt könnyedén elhagyhatjuk a currentsSmoker változót, mivel a másik változó tartalmazza a szükséges információt: 0 esetén nem dohányzik, ha az érték nagyobb mint 0, akkor igen. Hasonlóan járunk el a többi esetben is, arra ügyelve, hogy ahol az érték nem kérhető be könnyen a felhasználótól, azokat inkább elhagyjuk.

A vérnyomás pontos mérésére külön eszközre lenne szükség, illetve a prevalentHyp érték erősen korrelál a sysBP és a diaBP értékeivel, ezért utóbbi kettő változót elhagyom. Tehát a vérnyomás értéket az előrejelzésben az jelzi, hogy a páciens szenvedett-e magas vérnyomásban. A glukóz szint mérését szintén nem lehet elvégezni a meglévő okosóra eszközzel, továbbá korrelál a diabetes változóval, így a glucose oszlop is kikerül a tanító adatbázisból. A testtömegindex (BMI) a bekért magasság és testtömeg értékéből könnyedén kiszámítható, ezért az tartalmazni fogja az adatbázis.

Másik lépés, amit érdemes megtenni, hogy a függő változóval való korrelációt is megvizsgáljuk. A függő változó a TenYearCHD oszlop értékei, tehát a szívbetegség kockázatát jelző paraméter. Ez esetben például az education oszlop nem tűnik relevánsnak a modell szempontjából, ezért azt is eltávolíthatjuk. Emellett a koleszterinszint (totChol) is kikerül az adatbázisból az alacsony korreláció miatt, illetve azért, mert ennek az értéknek a mérése is külön eszközt igényelne, ami a tesztelést megnehezítené.

2.1.3. Eloszlás vizsgálata függő változón

Következő lépésben ellenőrzöm a függő változó eloszlását, ha az osztályok eloszlása kiegyensúlyozatlan, az hatással lehet a gépi tanulási modell teljesítményére. A kiegyensúlyozatlan adatok azt eredményezhetik, hogy a modell az egyik osztályt túlzottan preferálja, mivel az a domináns a tanító adatbázisban.



9. ábra:A célváltozó eloszlása

A diagram alapján megállapítható, hogy a célváltozó osztályainak eloszlása kiegyensúlyozatlan, mivel a 0 érték lényegesen nagyobb előfordulási gyakorisággal rendelkezik. A kiegyensúlyozatlan adatból fakadó problémát alul – és túlmintavételezés technikákkal lehet orvosolni. Ebben az elemzésben a túlmintavételezést választom, tehát a kisebbségi osztály (1-es osztály) mintáit növelem. Ehhez az imblearn Python könyvtár SMOTE (Synthetic Minority Over-sampling Technique) metódusát alkalmazom:

```
def smote(x,y):
    smote = SMOTE(random_state=42)
    X_res, y_res = smote.fit_resample(x, y)
    return X_res, y_res
```

A függvény bemeneti paraméterei az x (független változók) és az y (függő változó). A SMOTE objektum bemeneti értéke a random_state változó, amit 4-re állítok. Ha az érték például az általánosan használt 42, vagy egy másik konstans érték, akkor ugyanazt az eredményt kapom újra futtatás után. Amennyiben a random_state értékét nem adom meg, vagy 0-ra állítom, akkor minden futtatás során eltérő mintavételezési eredményeket kapnék. Legvégül a függvény a már átalakított adatokat adja tovább visszatérési értékként. [7]

2.1.4. Skálázás

Sok gépi tanulási modell függ attól, hogy a változóknak mi a mértékegysége. A skálázás hiánya torzíthatja a modellek teljesítményét, mert az különböző nagyságú változók hatása eltérően befolyásolják a predikciókat. A skálázás során az adatok úgy alakulnak át, hogy egy meghatározott tartományba kerüljenek. Ezt a műveletet többféle módon lehet elvégezni. Az egyik skálázási módszer a standardizálás, ebben az esetben ezt a megközelítést alkalmazom. A standardizálás során a változók úgy alakulnak át, hogy az eloszlásukat megtartják, de a változók középértéke 0, szórásuk pedig 1 lesz. Ehhez a scikit-learn gépi tanulási könyvtárának fogom használni, a StandardScaler függvényét.

```
def data_scaler(x_train, x_test):
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(x_train)
    X_test_scaled = scaler.transform(x_test)
    return X_train_scaled, X_test_scaled, scaler
```

2.2.Adattisztítás az alvási rendellenesség felismerésére szolgáló tanító adatbázisban

A továbbiakban a másik tanító adatbázis tisztítási műveleteire fogok fókuszálni, különös tekintettel azokra a pontokra, ahol eltérés tapasztalható az első feladathoz képest. A vizsgálatok

után megállapítható, hogy az adatbázis nem rendelkezik hiányos adatokkal, ezért nincs szükség adatpótlásra. Viszont a célváltozó None (nincs rendellenesség) értékét a Python a speciális konstansként értelmezte, amelyet a hiányzó vagy nem definiált értékek jelzésére használnak. Ez azonban eltér a szöveges adattól, ezért megfelelően kellett kezelni, hogy ne okozzon hibát.

Az alábbi függvény None szöveges karakterláncra cseréli hiányzó értéket:

```
data['Sleep Disorder'].fillna('None', inplace=True)
```

További probléma az adatokkal, hogy a testtömegindex (BMI) változó tartalmaz egyaránt Normal és Normal Weight kategóriát, amelyek megegyezőek. Ezeknek az összevonása indokolt, mégpedig úgy, hogy a Normal Weight kategóriát Normal értékre cseréljük.

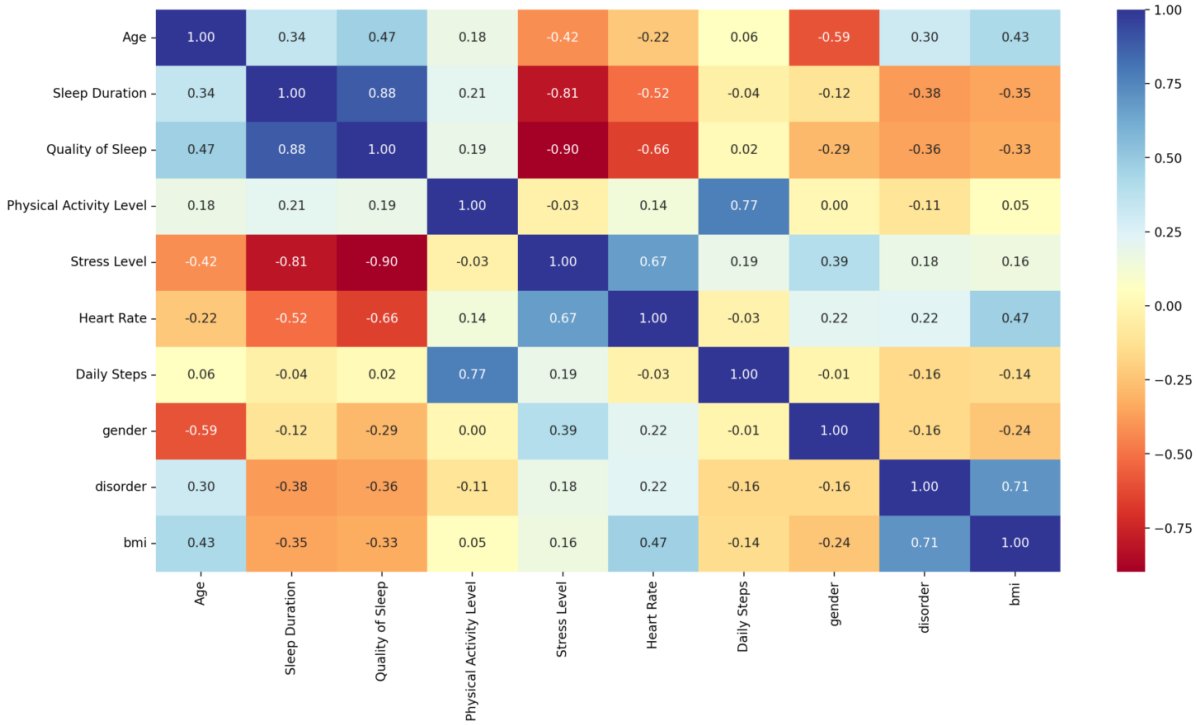
```
data['BMI Category'] = data['BMI Category'].replace({'Normal Weight': 'Normal'})
```

Az osztályok viszonylag kiegyensúlyozottak így a SMOTE metódus alkalmazása ezúttal nem indokolt. Az adatokat azonban szükséges skálázni, ez a lépés az első adatbázis esetében alkalmazott módszerrel megegyező módon történik. A korrelációs vizsgálatot érdemes külön elvégezni, mivel eltérő információkat nyújthat, amelyek befolyásolhatják a modell teljesítményét. Emellett a kategóriák kódolása is szükséges az adatokon.

2.2.1. A második tanító adatbázis korreláció vizsgálata

Ebben az adatbázisban jóval kevesebb változó található, ezért csak a legszükségesebb oszlopokat hagyom el, mivel a túl kevés változó negatívan befolyásolhatja a predikció pontosságát.

Az összefüggéseket az alábbi korrelációs mátrix tartalmazza.



10. ábra: Az alvási rendellenesség adatainak korrelációs mátrixa

A végső adatbázisból eltávolítjuk a foglalkozás (Occupation) és a vérnyomás (Blood Pressure) változókat. A foglalkozás változót azért hagyjuk el, mert az előrejelzések során inkább az egészségügyi adatokra fókuszálunk. A vérnyomás változót pedig az előző példával megegyező okból hagyjuk ki, mivel annak mérése külső eszközt igényel.

Az alvás minősége és az alvás hossza erősen korrelál egymással, illetve a napi lépés szám és az aktivitás szint között is szoros összefüggés figyelhető meg. A függő változó az rendellenesség (disorder), amellyel a testtömegindex (BMI) korrelál a leginkább. A pulzus és az életkor szintén nagy súllyal befolyásolhatja a modelleket. A negatív korreláció azt jelenti, hogy amikor az egyik változó értéke növekszik, a másik értéke csökken például, ha az alvás minőség alacsonyabb érték akkor magasabb a kockázata valamilyen alvási rendellenességnek.

2.2.2. Kategóriák kódolása

Az alvási értékeket tartalmazó tanító adatbázis bizonyos oszlopai szöveges értékeket tartalmaznak. Ezeket a szöveges adatokat mindenképpen át szükséges alakítani számmá, mivel a legtöbb gépi tanulási modell csak numerikus adatokat tud feldolgozni. Az átalakításhoz több különböző módszer léteznek, attól függően, hogy milyen típusú a szöveges adat. Az egyik, amit használok az a Label Encoding (címkézés) technika. Ezt a technikát akkor szokás alkalmazni, ha viszonylag kevés kategória van.

A példámban a nem (Gender) oszlop szöveges értékeit szükséges átalakítani, 0 és 1 értékre. A korábban már többször használt scikit-learn könyvtár LabelEncoder függvényét használom az átalakításhoz:

```
label_encoder_gender = LabelEncoder()
data['gender'] = label_encoder_gender.fit_transform(data['Gender'])
```

Egy másik alkalmazott technika az Ordinal Encoder, ami például akkor használható, amikor a kategóriák között van sorrend. Például a BMI (testtömeg-index) oszlop esetén, ahol a kategóriák:

- Normal (normális testalkat)
- Overweight (túlsúlyos)
- Obese (elhízott)

A BMI értékeit tehát az Ordinal Encoder segítségével alakítom át. Továbbá a függő változó (disorder) kategóriáival is ugyanígy járnuk el, amelyek szintén szöveges értékek. Az alábbi kód ezeket az átalakításokat mutatja be.

```
encoder_disorder = OrdinalEncoder(categories=[['None', 'Sleep Apnea', 'Insomnia']])
encoder_bmi = OrdinalEncoder(categories=[['Normal', 'Overweight', 'Obese']])
data['disorder'] = encoder_disorder.fit_transform(data[['Sleep Disorder']])
data['bmi'] = encoder_bmi.fit_transform(data[['BMI Category']])
```

Ez a két technika segít abban, hogy a szöveges adatokat numerikus formába alakítsam, amit a gépi tanulási modellek könnyen fel tudnak dolgozni.

3. Machine Learning modellek

A következő fejezetben az általam használt Machine Learning modelleket fogom bemutatni. A gépi tanulást a mesterséges intelligencia egyik ágának tartják, amelynek a célja olyan modellek létrehozása, amelyek képesek adatokból tanulni és következtetéseket levonni közvetlen utasítás nélkül. A gépi tanulás során az algoritmusok mintákat fedeznek fel, amelyek alapján képesek predikciókat végezni, vagy döntéseket hozni.

Három fő gépi tanulási módszer létezik:

- Felügyelt tanulás: A modellnek előre meghatározott bemeneti és kimeneti párokat adunk.
- Felügyelet nélküli tanulás: Az adatoknak nincs címkézett kimenete, és a modell célja az adatok közötti mintázatok felismerése.
- Megerősítő tanulás: A modell különböző döntéseket hoz, és ezekért jutalmazva, vagy büntetve van.

A predikció végzésére felügyelt tanulási módszereket alkalmazok mindkét tanító adatbázis adataira. Felügyelt tanulás lényege, hogy az adatokat címkézzük, majd a modellt betanítjuk ezen adatok alapján, hogy hatékonyan tudjon döntéseket hozni és előrejelezni. Miután megtörtént az adatok tisztítása és feldolgozása, a következő lépés a modellek betanítása. Ehhez kiválasztom a függő és a független változókat, majd az így előkészített adatokat két csoportra bontom, betanítási és tesztelési adathalmazra. Az adatok szétválasztásához a scikit-learn könyvtár train_test_split függvényét használom.

```
def split_data(X_res, y_res):
    x_train, x_test, y_train, y_test = train_test_split(X_res, y_res, test_size = 0.2, random_state = 42)
    return x_train, x_test, y_train, y_test
```

A függvény megkapja bemeneti paraméterként a már feldolgozott és felosztott adatokat. A test_size = 0.2 beállítja, hogy az adatok 20%-át tesztelésre, míg 80%-át tanulásra használjuk. A random_state értékének megadása biztosítja az eredmények reprodukálhatóságát, ahogy korábban már részleteztem. Az programban öt különböző modellt alkalmazok, amelyek a már szétosztott adatokból tudják elvégezni a predikciókat. [8]

3.1. Logisztikus regresszió

A logisztikus regresszió egy osztályozási algoritmus, amelyet elsősorban bináris problémák megoldására alkalmaznak. Az algoritmus az adatok közötti összefüggések modellezésével segít előre jelezni a kategóriákat. A használatának az előfeltételei:

- A változók között nincs multikollinearitás.
- A független változók lineárisan kell utaljanak a függő változóra.
- Nem követeli meg a normális eloszlást.
- Nem feltételez linearitást a függő és a független változók között.

A logisztikus regresszió kimenete nem kategória, hanem egy valószínűség, amely 0 és 1 között mozog. Az erős valószínűségek megközelítik az egyet, míg a gyenge valószínűségek a 0-t, ami arra utal, hogy az esemény bekövetkezésének az esélye nagyon alacsony. A logisztikus regresszió az úgynevezett logit függvényt használja, aminek a kimenetét egy logaritmikus skálán ábrázolja. Ez azt jelenti, hogy az esélyhányados egyenlő a siker valószínűsége (p) osztva a sikertelen valószínűséggel (1-p). Az esélyhányados logaritmusával számolunk a logisztikus regresszió egyenletében:

$$logit(p) = \log\left(\frac{p}{1-p}\right)$$

Az eredményt egy logisztikus függvény segítségével alakul vissza valószínűségi értékké, amely 0 és 1 között mozog.

A Logisztikus Regresszió egy gyors, hatékony és egyszerű modell. Azonban, ha az adatok nem lineárisak, akkor a modell nem képes pontos előrejelzéseket végezni. Továbbá a modell érzékeny az adathalmaz kiugró értékeire is. [9]

3.1.1. Logisztikus regresszió alkalmazása

Az előrejelzésekhez a scikit-learn LogisticRegression függvényét fogom használni. A modell az alábbi függvényben végzi el a számításokat, miután megkapta bemeneti paraméterként szétosztott adatokat.

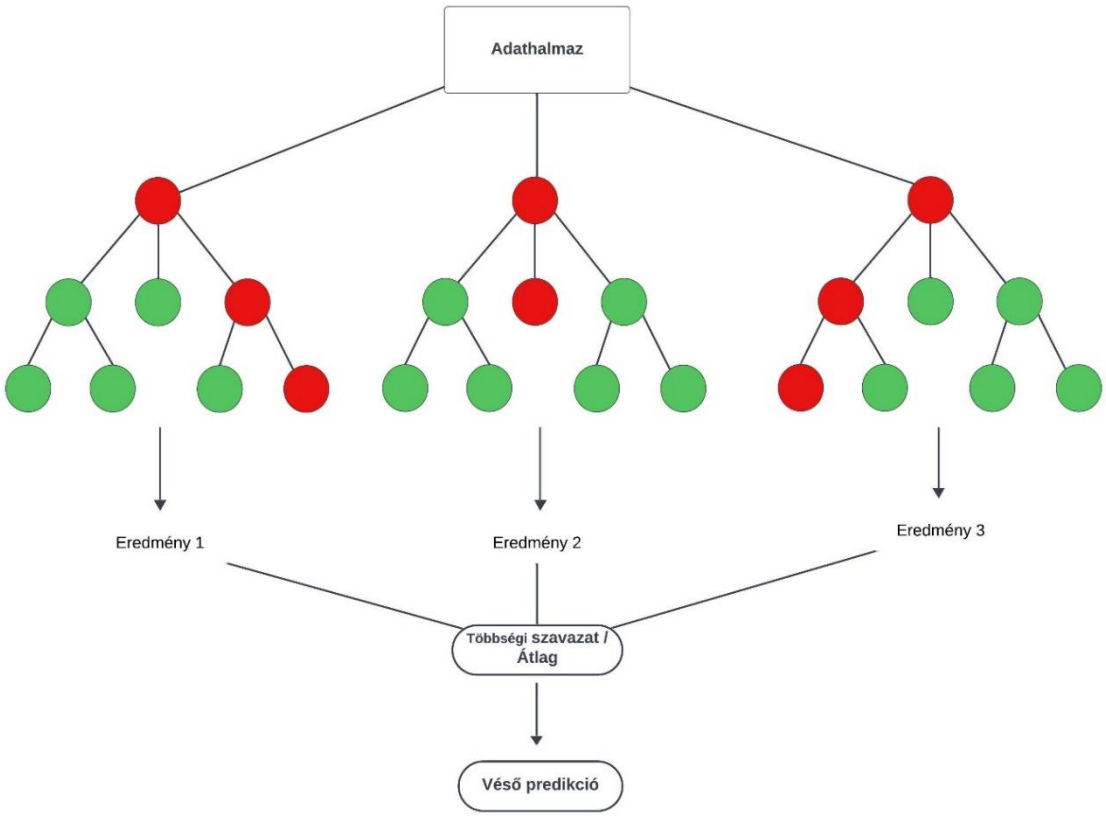
```
def train_log_reg(X_train_scaled, y_train):
    LogRegModel = LogisticRegression()
    LogRegModel.fit(X_train_scaled, y_train.values.ravel())
    return LogRegModel
```

Különböző bemeneti paraméterekkel lehet beállítani a modellt, Amelyek segítségével optimalizálható a modell teljesítménye. Ha nem állítok be semmit, akkor alapértelmezett paramétereket kapja a modell bemenetként. Ebben az esetben maradtak az alapértelmezett beállítások, amelyek többnyire megfelelőnek bizonyultak mindkét predikció esetében, mivel a teszteléseim során más bemeneti értékekkel sem kaptam szignifikánsan eltérő eredményt a modell teljesítményére. A célváltozót (y_train) még átesik egy átalakításon, mielőtt megkapja a modell az adatokat. Az átalakítás egy Numpy függvény segítségével történik és a

végeredmény egy egydimenziós tömb lesz. Erre azért van szükség, hogy a célváltozó kompatibilis legyen a gépi tanulási modell betanítási folyamatával.

3.2.Random Forest

A Random Forest (véletlen erdő) egy felügyelt gépi tanulási módszer, amely egy predikciót ad kimenetként. A véletlen erdő elnevezés onnan ered, hogy a modell több döntési fa eredményét együttesen veszi figyelembe. Minden fa a bemeneti adatoknak egy véletlenszerű részhalmazát kapja és az egyes változók közül is véletlenszerűen választ ki egy részt, hogy csökkentse a túlilleszkedést az adatokra. A modell alkalmazható regressziós és osztályozási feladatokra is. Regressziós problémák esetén fák által előre jelzett értékek átlaga lesz a végső becslés. Osztályozási feladatoknál a legtöbb döntési fa által javasolt osztály adja a végső kimenet.



11. ábra: Random Forest

A Random Forest alapvetően nagy pontosságú és megbízhatóságú algoritmus. Hátránya azonban, hogy erőforrás- és időigényes a számítás folyamata. Több erőforrásra azért van szüksége az algoritmusnak, mert nagy adathalmazokat dolgoz fel. Továbbá időigényes, mivel minden külön döntési fa eredményéhez külön számítást végez. Azonban különböző paraméterek megadásával lehetőség van optimalizálni a modellt.

3.2.1. Random Forest alkalmazása

A Random Forest modellhez a scikit-learn RandomForestClassifier függvényét alkalmazom. Az alábbi függvényt tartalmazza a beállításokat mindkét előrejelzés esetében egyaránt.

```
def train_random_forest(X_train_scaled, y_train):
    RFModel = RandomForestClassifier(n_estimators=200, random_state=42, max_depth=
    80, min_samples_split= 8)
    RFModel.fit(X_train_scaled, y_train.values.ravel())
    return RFModel
```

Az első megadott paraméter az a n_estimators, ami a döntési fák számát határozza meg. Minél több fa van, annál jobb lehet a modell teljesítménye, viszont ez növelheti a számítási időt. A saját predikcióimhoz 200 döntési fát állítottam be, ami több mint az alapértelmezett 100 fa, viszont így is korlátozva van a számítási idő valamennyire. A max_depth értékkel megadom a döntési fák maximális mélységét. Ez segít abban, hogy elkerüljük a túlilleszkedést vagyis, hogy a fa több apró részletet tanuljon meg, például még a zajokat is. A min_samples_split paraméter a minimális számú minták száma. Minél magasabb ez az érték, annál mélyebbek lesznek a fák. Itt az alapértelmezett 2-nél nagyobb értéket adtam, szintén a túlilleszkedés megelőzése céljából. A random_state esetében pedig ezúttal is a modell eredményeinek reprodukálhatósága miatt lett megadva. [10]

3.3.Boosting

A boosting egy olyan gépi tanulási módszer, amely az úgynevezett gyenge tanulók sorozatos kombinálásával hoz létre egy jobban teljesítő prediktív modellt. A gyenge tanuló modellek, például döntési fák, amelyeknek a teljesítménye kicsit jobb, mint a véletlenszerű találat. A boosting során minden egyes modell megpróbálja kijavítani az elődje hibáit. Ennek a technikának több változata létezik:

- AdaBoost
- XGBoost
- Gradient Boosting

A saját predikciós feladataimhoz az XGBoost módszert fogom használni. Az XGBoost a Gardian Boosting algoritmus egy továbbfejlesztett változata. Ez a technika nagyon gyors, jól teljesítő és hatékony nagy adathalmazokon. Viszont kis adathalmazokon előfordulhat pontatlan eredmény, mivel túlságosan összetett az algoritmus. [11]

3.3.1. XGBoost alkalmazása

Az XBoost a DMLC (Distributed Machine Learning Community) által fejlesztett algoritmus. A következő függvény tartalmazza az XGBoost modellt, amihez a DMLC XGBoost python könyvtárat alkalmazom.

```
def train_xgboost(X_train_scaled, y_train, is_chd):
    if is_chd == True:
            XGBModel = xgb.XGBClassifier(colsample_bytree = 0.8,
        objective='binary:logistic', eval_metric = 'logloss', n_estimators = 200, learning_rate
        = 0.4, max_depth = 5, subsample = 0.8)
    else:
            XGBModel=xgb.XGBClassifier(colsample_bytree=0.8,
        objective='multi:softmax', eval_metric = 'mlogloss', n_estimators = 200,
        learning_rate = 0.4, max_depth = 5, subsample = 0.8)
    XGBModel.fit(X_train_scaled, y_train.values.ravel())
    return XGBModel
```

A colsample_bytree paraméter határozza meg, hogy a döntési fák építése során az egyes fák minden egyes csomópontjához milyen arányban használja fel a bemeneti adat tulajdonságait. Alapértelmezetten ez az érték 100 %, a példámban 80 %-ra állítottam. Ezzel csökkenthetem a túlilleszkedés esélyét, emellett kevesebb erőforrást igényel a modell futtatása.

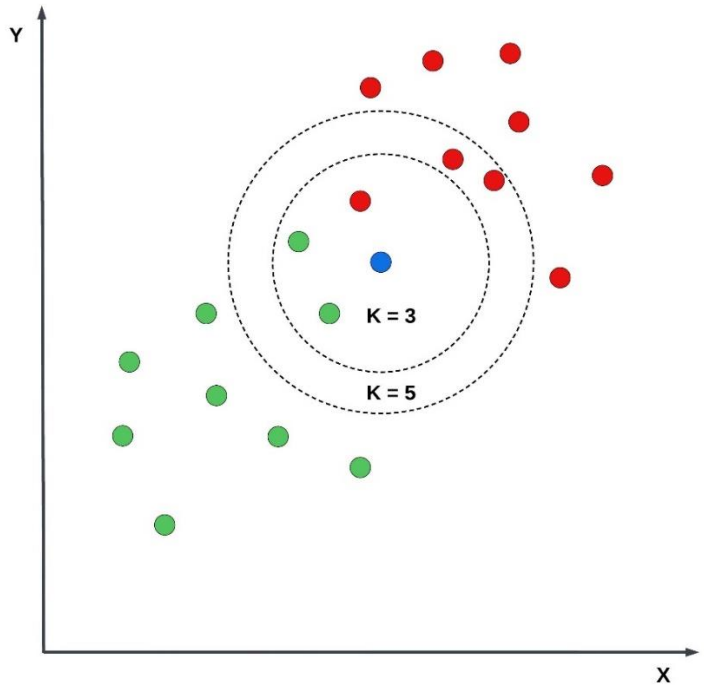
A kardiovaszkuláris probléma kockázatának predikciója egy bináris osztályzási feladat, az ehhez tartozó funkciókat az objective 'binary:logistic' értékével állítom be. Az eval_metric pedig az értékelési metrika a modell teljesítményének mérésére.

Az alvási rendellenesség vizsgálat egy több változós osztályzási feladat, ezért objective='multi:softmax' és eval_metric='mlogloss' paraméterekkel dolgozunk. A learning_rate a tanulási ráta, amit magasabbra állítottam az alapértelmezett beállításnál, mivel így gyorsul a tanulási folyamat és jobb lehet a modell teljesítménye is. A subsample az adatok részhalmazának aránya, amit a fák létrehozásakor használ a modell. Ebben az esetben a minták 80%-át használja. A döntési fák száma, illetve a maximális mélység beállítását már ismertettem a Random Forest fejezetben.

3.4.K-Nearest Neighbors

A K-Nearest Neighbors egy olyan gépi tanulási technika, amely egyaránt használható osztályozó és regressziós feladatokra. A módszer azon az elven alapul, hogy a hasonló adatpontok általában hasonló címkékkel vagy értékekkel rendelkeznek. A tanulási szakasz során a KNN algoritmus a teljes képzési adathalmazt eltárolja. Az előrejelzések készítésekor kiszámítja a bemeneti adatpont és az összes képzési példa közötti távolságot egy választott

távolsági módszer segítségével. Ezután az algoritmus a bemeneti adatpont K legközelebbi szomszédját azonosítja a távolságuk alapján. A K érték a szomszédok számát jelenti.



12. ábra: K-Nearest Neighbor

Osztályozási feladat esetén az új adatpont osztályát úgy határozzuk meg, hogy a leggyakoribbat vesszük a K legközelebbi szomszéd osztályaiból, míg Regresszió esetén az új adatpont értéke a K legközelebbi szomszéd értékeinek átlaga lesz. Végül a KNN algoritmus súlyozza a szomszédokat például a távolságuk alapján, vagyis a közelebbi szomszédok nagyobb súllyal befolyásolják a döntést.

A KNN egy egyszerű algoritmus, mivel nem igényel túl sok beállítható paramétert, emellett általában jól teljesít kisebb adathalazokon is. A hátránya, hogy rendkívül számításigényes, ezáltal a memóriaigény is magasabb.

3.4.1. KNN alkalmazása

A KNN gépi tanulási modellhez a sklearn.neighbors könyvtár KNeighborsClassifier függvényét használom.

```
def train_KNN(X_train_scaled, y_train):
KNN = KNeighborsClassifier(n_neighbors=3, weights='distance', metric='manhattan' )
KNN.fit(X_train_scaled, y_train.values.ravel())
return KNN
```

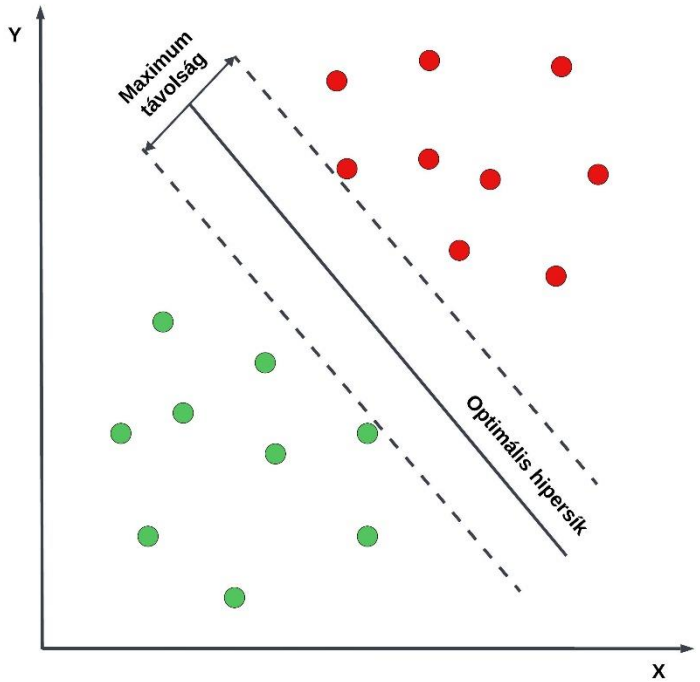
Az n_neighbors változó adja meg a legközelebbi szomszédok számát, amit figyelembe vesz az algoritmus az osztályzás során. Ebben az esetben a számításhoz 3 legközelebbi szomszéd

lett beállítva. A döntéshozatalhoz a szomszédok hozzájárulásának súlyozását a weights paraméterrel határozzuk meg. A függvényben beállított distance érték azt jelenti, hogy a közelebbi szomszédok nagyobb súlyt kapnak, a távolságokkal fordított arányban. Mivel a közelebbi szomszédok általában nagyobb relevanciával bírnak, ez a módszer segíthet pontosabb predikciókat adni, ha nem egyenletes az adatok eloszlása. A metric beállítja a távolságmérés módszerét, ami tartalmazza a legismertebb távolságmérő metrikákat, mint például a Manhattan és az Euklideszi távolságot. Ezúttal a Manhattan-távolságot alkalmazom, mivel ennek a módszernek gyorsabb a számítása. [12]

3.5.Support Vector Machine

Az utolsó általam alkalmazott technika a Support Vector Machine (SVM). Ez egy gyakran használt gépi tanulási technika, mivel viszonylag alacsony számítási teljesítmény mellett magas pontosságra képes. Az SVM algoritmus célja, hogy találjon egy olyan hipersíkot, amely alapján elválaszthatóak különböző osztályok tagjai egymástól úgy, hogy azok a lehető legnagyobb távolságban legyenek egymástól. Ha nem létezik ilyen hipersík, akkor átalakítja a tanító adatokat, majd ezután keresi meg az optimális elválasztó hipersíkot.

Az SVM főleg kisebb adathalmazokon hatékony, illetve a túlilleszkedés kockázata is csökkenthető a távolság növelésével. Nagy adathalmazokon viszont nagy memóriaigényű és időigényes lehet a számítás. [13]



13. ábra: Support Vector Machine

3.5.1. SVM alkalmazása

Az SVM modell előrejelzését ezúttal is az sklearn könyvtár SVC függvényének segítségével végzem el.

```
def train_SVM(X_train_scaled, y_train):
    SVMModel = svm.SVC(probability=True, C=10)
    SVMModel.fit(X_train_scaled, y_train.values.ravel())
    return SVMModel
```

A propability (valószínűség) beállítással megadom, hogy valószínűségi becslést is számítson a modell. Ez nagyobb memóriahasználatot és számítási időt igényel, viszont növelheti a pontosságot. A C a regularizációs paraméter, amely meghatározza a hibás osztályozások megengedett mennyiségét. Kisebb C érték szigorúbb modellt eredményez, amely megpróbál minden adatpontot jól osztályozni, viszont növeli a túlilleszkedés esélyét. Ezért az alapértelmezett 1 helyett magasabb értéket állítottam be, amely jelen esetben 10.

4. Eredmények összehasonlítása, elemzése és végkövetkeztetés

Miután elkészültek a Machine Learning modellek, a következő lépés az eredmények összehasonlítása és elemzése. Az értékelés során a szétosztott adatok tesztelésre szánt részét használjuk. Több érékelési metrika létezik a teljesítmény vizsgálatára, feladathoz az alábbi módszereket alkalmazom:

- Accuracy (Pontosság)
- Precission (Precízitás)
- Recall (Visszahívás)
- F1 Score
- ROC-görbe

Ezeknek a metrikáknak az eredményei alapján fogom kiválasztani, hogy melyik megközelítés a legoptimálisabb az adott predikciós feladathoz.

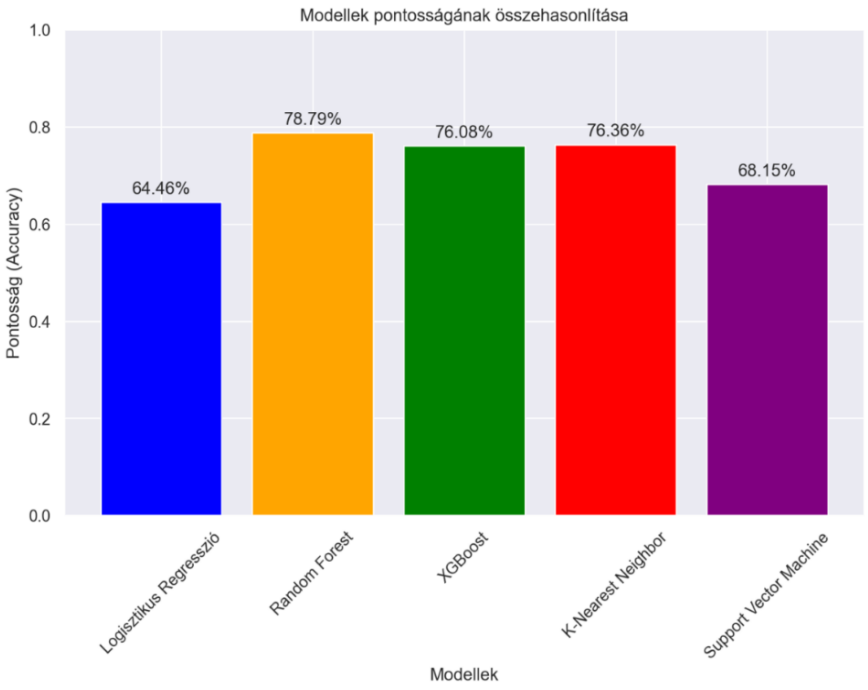
4.1.Pontosság

Az accuracy (pontosság) az egyik leggyakrabban használt értékelési metrika a teljesítmény vizsgálatára. A pontosság megmutatja, hogy a modell által hozott előrejelzések közül mennyi volt helyes, azaz az összes esetből hányat sikerült helyesen osztályozni. A mérés hasznos kiegyensúlyozott eloszlású adatokon, viszont kiegyensúlyozatlan adatok esetén megtévesztő lehet, mivel egy domináns osztály esetében akkor is előfordulhat magas accuracy érték, ha csak a gyakori osztályokat osztályozza helyesen. [14]

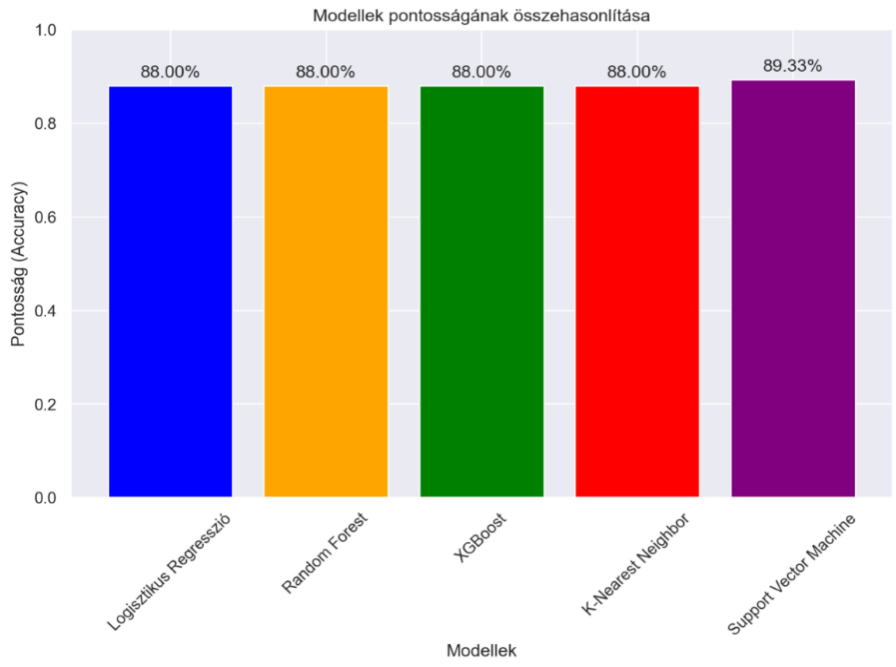
A pontosság számításának a képlete a következő:

$$Accuracy = \frac{Helyes\ előrejelzések\ száma}{Összes\ előrejelzés\ száma}$$

A két feladat modelljeinek eredményeit egyszerű oszlopdiagramokon fogom szemléltetni.



a



b

14. ábra: Modellek pontossága

a – Szívkoszorúér feladat; b – Alvási rendellenesség feladat

Az első esetben minden mérésnél 80% alatti az eredmény kapunk, ami közepesnek mondható. Emellett ebben a feladatban a célváltozó osztályai kiegyensúlyoztam, így további metrikák eredményeire is szükség lesz, hogy a teljesítményt megfelelően értékelni lehessen.

A második példánál már egységesen 88% körüliek az eredmények, ami már kifejezetten jónak számít. Azonban megállapítható, hogy mindegyik módszer közel azonos szinten teljesített. Ilyekor előfordulhat, hogy a tanító adataink nem megfelelő minőségűek, vagy egyszerűen nem túl összetett az elvégzendő feladat a modellek számára. Ebben az esetben is szükséges további vizsgálat a megfelelő következtetések levonásának érdekében.

4.2.Precission, Recall, F1 score

Mivel az accuracy érték nem ad teljes képet a teljesítményről, ezért szükség van más megközelítésekre, ahol részletesebben lehet megismerni az eredményeket. Erre alkalmas metrikák: precission (precízió), recall (visszahívás), F1 score. Amely metrikák számításához az osztályozási feladatok kimeneteinek értékelésében használt alapvető kategóriákat használunk. Amiket az alábbi táblázat mutat be:

3. táblázat: Alapvető kategóriák

Kategória	Definíció
True Positive (TP) – Valódi pozitív	*A modell helyesen azonosított egy pozitív esetet*
True Negative (TN) – Valódi negatív	*Helyesen azonosított egy negatív eseteket*
False Positive (FP) – Hamis pozitív	*Tévesen pozitívként azonosít egy valójában negatív esetet*
False Negative (FN) – Hamis negatív	*Tévesen negatívként azonosít egy valójában pozitív esetet.*

Precission esetében a fő cél a téves pozitív előrejelzések elkerülése. A számításához használt képlet:

$$Precission = \frac{TP}{TP + FP}$$

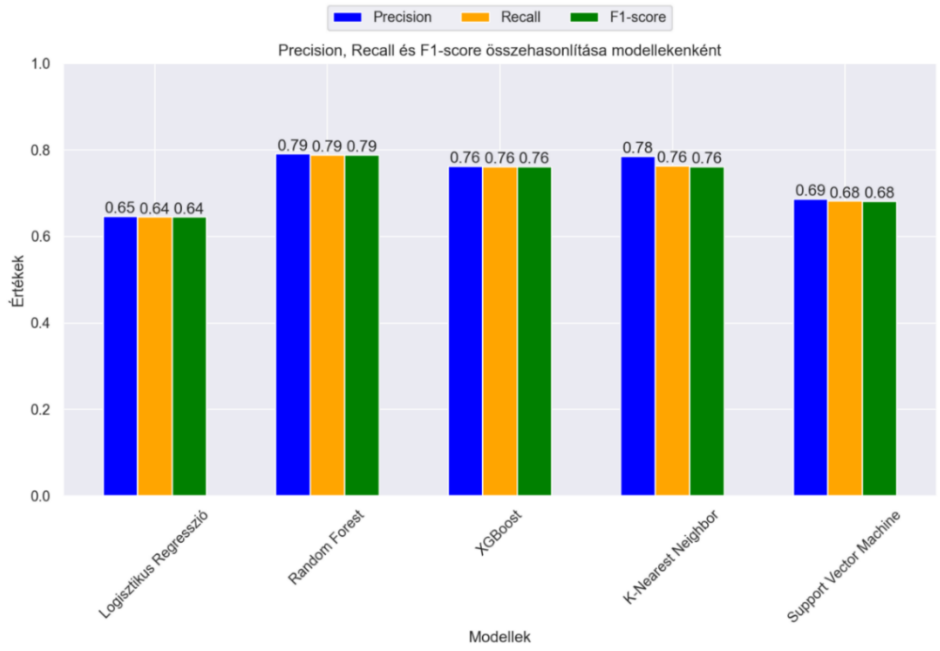
A recall számítása azért fontos, hogy megtudjuk a tényleges pozitív esetek közül mennyit sikerült helyesen felismernie a modellnek, vagyis a téves negatívok minimalizálása. Az eredményt a következőképpen kapjuk:

$$Recall = \frac{TP}{TP + FN}$$

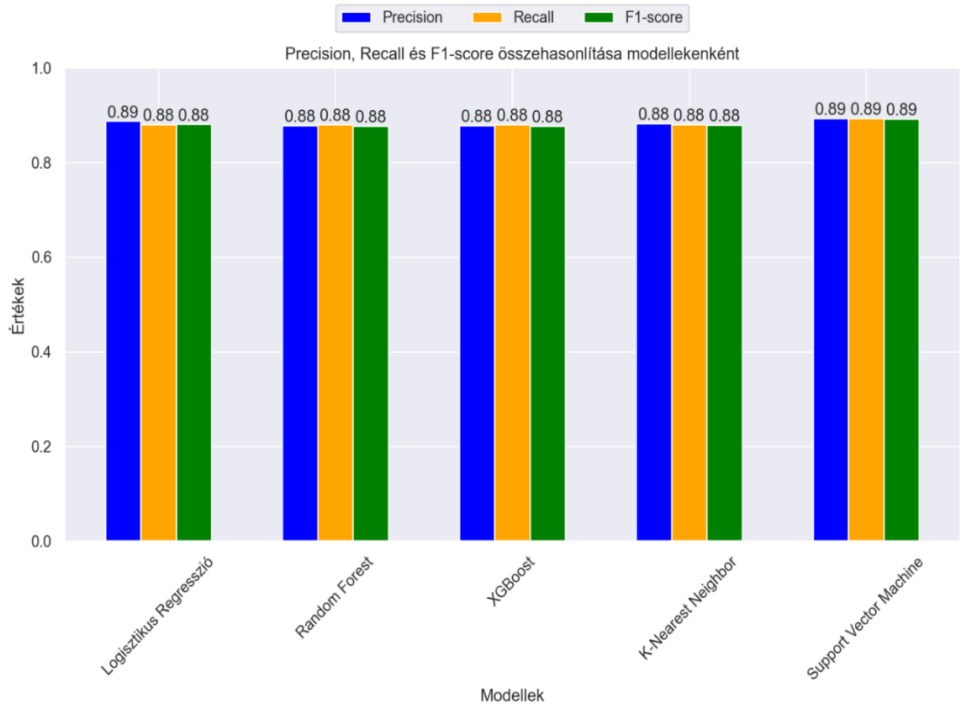
Az F1-score a Precision és Recall harmonikus átlaga, amivel az egyensúlyt méri. Ez a metrika akkor hasznos, ha fontos, hogy mindkét mérőszámot figyelembe vegyük. Képlete:

$$F1 = 2\,x\,\frac{Precission\ x\ Recall}{Precission + Recall}$$

A F1-score a Precision és a Recall végeredményeit szintén kiválóan lehet oszlopdiagrammal szemléltetni.



a



b

15. ábra: F1-score, Precision és a Recall

a – Szívkoszorúér feladat; b – Alvási rendellenesség feladat

Az alvás rendellenesség predikciónál látható, hogy mindegyik modell mindegyik értéke közel azonos. A korábbi megállapítás így megerősítést nyer, hogy a tanító adatok nem elég

változatosak, továbbá az adat mennyisége kevés. Így a modelleknek viszonylag könnyen érhet el magas eredményt.

A másik feladatnál viszont már jóval változatosabbak a diagramok. Az XGBoost, a KNN és a legjobban teljesítő, amíg a Random Forest kiemelkedik a többi modell közül. A Recall metrika különösen fontos ebben az esetben, mivel egészségügyi adatokból betegségek kiszűrése a modellek célja. A fő feladat tehát a téves negatív esetek minimalizálása. Betegség diagnosztika esetében ez azt jelenti, hogy tévesen egészségesnek azonosítunk egy valójában beteg pácienst. Megállapítható tehát, hogy ebben az összehasonlításban egyértelműen a Random Forest teljesít a legjobban. [15]

4.3.ROC-görbe

A ROC-görbe (Receiver Operating Characteristic Curve) egy grafikus ábrázolás, amely a modell különböző küszöbértékei mellett mutatja a hamis pozitív és a valódi pozitív arányt.

4. táblázat: Hamis pozitív és valódi pozitív arány

Arány	Képlet
Hamis Pozitív Arány	$FPR = \frac{FP}{FP + TN}$
Valódi Pozitív Arány (FPR)	$TPR = \frac{TP}{TP + FN}$

Egy másik érték, ami megállapítható ebből a vizsgálatból az a görbe alatti területet mérő AUC (Area Under the Curve) érték. Amely egy összefoglaló érték arról, hogy a modell mennyire képes megkülönböztetni egymástól a pozitív és a negatív osztályokat. Az AUC értéke 0,5 és 1 között mozog, ahol 1 jelenti a tökéletes osztályozást, 0,5 pedig a véletlenszerű találgatást.

4.3.1. ROC-görbe bináris osztályzás esetén

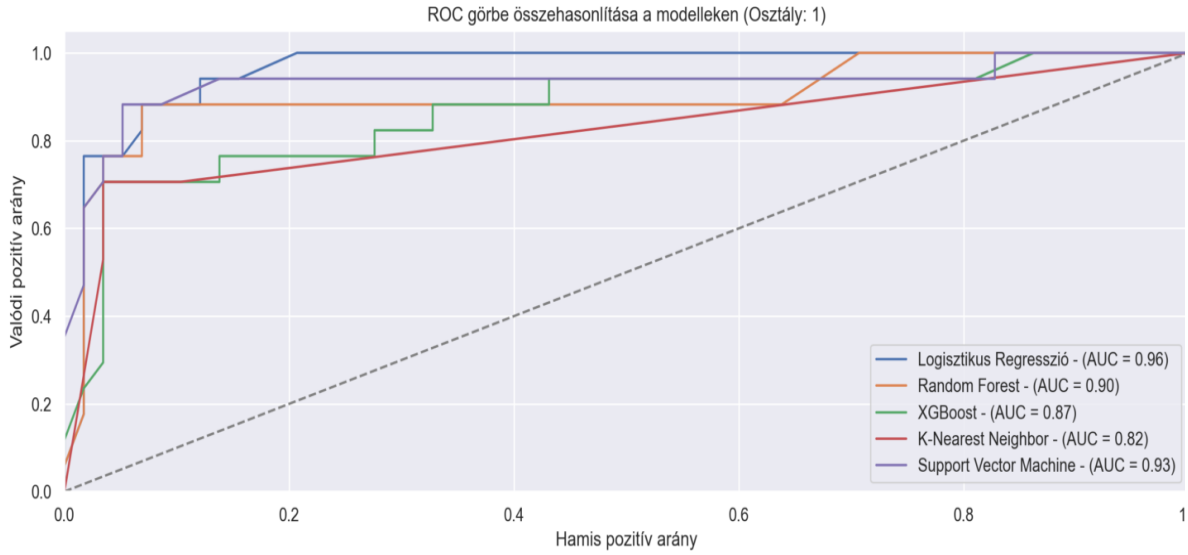
A kardiovaszkuláris kockázat predikció bináris osztályzási feladat, ezért a ROC-görbe ebben az esetben egyszerűen használható.

16. ábra: ROC-görbe bináris osztályzás

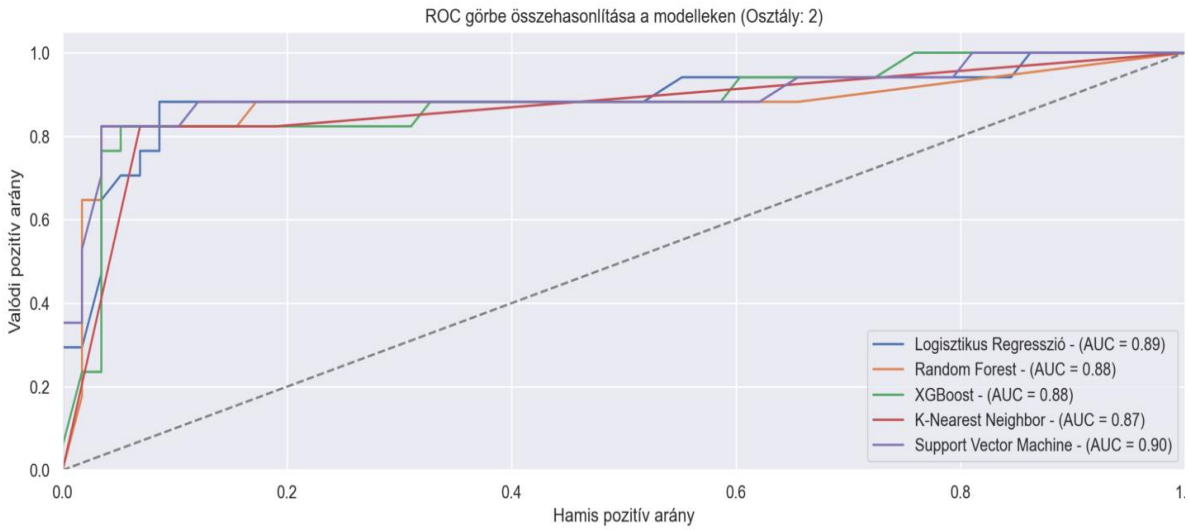
A középen húzott szaggatott vonal a véletlenszerű osztályozást jelképezi, míg a többi vonal a különböző modellek ROC görbéi. Az AUC értékéből megállapítható, hogy ezúttal is a Random Forest a legmegbízhatóbb ennél a feladatnál. Az XGBoost és a KNN ér még el magasabb pontszámot. Az egészségügyi predikciók során a küszöbérték beállítása gyakran változhat, attól függően, hogy a cél a betegségek felismerése vagy a téves riasztások minimalizálása. Az ideális küszöbértéket a legegyszerűbben úgy lehet megállapítani, hogy megkeressük a bal felső sarok és a görbe pontjai közül mérhető legkisebb távolságot. Az általam végzett előrejelzésekben a küszöbértéknek 0.5 lett beállítva, ami az alapértelmezett küszöbérték, de a tényleges alkalmazástól függően szükség lehet a küszöb módosítására.

4.3.2. ROC-görbe többosztályos feladat esetén

A ROC-görbe alapvetően bináris osztályzási feladatok esetén használható egyszerűen. Ha több osztály van, akkor más megközelítést kell alkalmazni. Az egyik ilyen megközelítés a One vs Rest (OvR). Ilyenkor a többosztályos problémát több bináris problémára kell felbontani, és ROC-görbét rajzolni minden osztályra úgy, hogy az adott osztály legyen a pozitív és a többi a negatív. Tehát az alvási rendellenesség detektálás során külön vizsgáljuk meg az alvási apnoé (osztály: 1) és az insomnia (osztály: 2) osztályokat.



a



b

17. ábra: ROC-görbe többosztályos feladat

a – Alvási apnoé; b – Insomnia

Az első diagramról leolvasható, hogy a Logisztikus Regresszió és az SVM kiváló 0.9 feletti eredményt ért el. A Random Forest is elég jól teljesít, az XGBoost és KNN valamelyest gyengébben, viszont még elfogadható mértékben.

A második osztály esetében is a Logisztikus Regresszió és az SVM teljesített a legjobban minimálisan, de itt az összes különbség szinte elhanyagolható. [16]

4.4.Végkövetkeztetések

A dolgozatom egyik célja az volt, hogy a gépi tanulási modellek teljesítményeinek az összehasonlítása után kiválasszam a legmegfelelőbbet, amit végül az alkalmazásban használok a predikció végzésére. Miután több mérési metrikát alkalmaztam, ezáltal megalapozottabban tudtam elvégezni a kiválasztást.

A kardiovaszkuláris probléma kockázatának előrejelzése feladat összehasonlítása után egyértelműen megállapítható, hogy a Random Forest a legjobban teljesítő módszer. Mivel a Random Forest teljesít a legjobban mind az öt általam használt metrika esetében. Így ehhez a feladathoz ezt a modellt fogom használni az elkészült alkalmazásban.

Az alvási rendellenesség detektálás feleladatnál már nem volt ennyire egyértelmű a választás, mivel itt valamennyi módszer egyaránt magas eredményeket ért el. Nagyon minimális különbséggel az SVM modell teljesített a legjobban. Az Accuracy, Precission, Recall és az F1 score értéke is ennek a modellnek a legmagasabb, emellett a ROC-Görbe alapján is ott volt a legmagasabb értéket elérő módszerek között. Viszont a különbség annyira elhanyagolható, hogy további manuális tesztelés elvégzésére volt szükség.

A tesztelés során az SVM bizonyos bemeneti paraméter kombináció esetén, nem adott minden esetben logikus és konzekvens eredményt. Miután elvégeztem a többi modellen is a maunuális tesztelést összeségében a KNN esetében tapasztaltam a legkevesebb anomáliát. Így az alkalmazásomban az alvási rendellenességet előrejelző feladathoz a K-Nearest Neighbor módszert fogom alkalmazni.

4.5.Fejlesztési lehetőségek

Az elkészült alkalmazás továbbfejlesztésére és a modellek eredményeinek javítására egyaránt lehetőség van. Jelenleg csak a Garmin gyártó okosóráinak az adatainak a lekérése támogatott, azonban más gyártók API szolgáltatásait is lehetne integrálni, így sokkal több felhasználó tudná a saját adatait felhasználni a predikciókhoz.

A gépi tanulási modellek több módon is fejleszthetőek. Az adatminőség javítása és az adatmennyiség növelése fontos szerepet játszik a modellek teljesítményének javításában. Emellett a modelleket számos különböző paraméterrel lehet ellátni, ebből adódóan elképzelhető, hogy eltérő beállításokkal jobb teljesítményt lehetne elérni bizonyos modelleknél. Továbbá az adattisztítási folyamatok optimalizálásával is növelhető a pontosság.

Jelenleg az alkalmazásban öt különféle felügyelt gépi tanulási módszer kerül bemutatásra, de ezen felül számos más megközelítés is létezik, amelyek alkalmazásával eltérő és akár jobb végeredmények is elérhetők lennének.

4.6.Az alkalmazás futtatása és erőforrás szükséglete

4.6.1. Megosztás a Streamlit cloud platformon

Az utolsó lépésben az alkalmazás megosztásra került a Streamlit cloud platformon, hogy nyilvánosan elérhetővé váljon. A feltöltés előtt szükséges volt az adatbázist is egy felhőalapú szolgáltatóhoz áthelyezni, mivel eredetileg a localhost környezetben futott. Az adatbázis tárolására a Supabase szolgáltatását választottam, amely lehetővé teszi a PostgreSQL alapú adatbázisok rugalmas és biztonságos felhőalapú kezelését ingyenesen. Viszont így az adatbázis kapcsolat jóval lassabban jött létre, mint a localhost esetében. Ezért az adatbázis műveleteinek optimalizálására volt szükség.

Az egyik megoldás az adatok tömeges betöltése volt, amely lehetővé tette nagy mennyiségű adat egyszerre történő betöltését. Ahelyett, hogy az adatokat egyenként illesztenénk be az adatbázisba, a tömeges betöltés során több adatot egyszerre kezelünk, ami csökkenti az adatbázisba történő beillesztési időt és a hálózati késleltetést. Továbbá az SQL lekérdezések dinamikus létrehozása is hozzájárult a gyorsabb adattovábbításhoz. Ahol egy metódus az egyes értékeket dinamikusan formázza SQL formátumúvá, majd ezek egy változóban lettek összefűzve, hogy egyetlen lekérdezésként adhassuk át az adatokat. Ezekkel a módosításokkal már jelentősen gyorsabb lett az alkalmazás futás közben.

Szintén gondoskodni kellett a feltöltés során a függőségekről. Az alkalmazásban használt összes könyvtár a projektmappában található requiremet.txt állományban vannak felsorolva. Ez a fájl meghatározza, hogy melyik csomagokra van szüksége az alkalmazásnak a megfelelő működéshez. A Streamlit Cloud automatikusan beolvassa ezt a fájlt, és megpróbálja telepíteni az abban meghatározott könyvtárakat. Ha valamelyiknek a verziója nem kompatibilis az alkalmazás futtatásához használt környezettel, vagy ha a függőségek között verzióütközés van, hibák léphetnek fel. Ezért a nem használt csomagokat eltávolítottam, illetve nem adtam meg verziószámot, így végül kiküszöböltem a kompatibilitásból fakadó problémákat. Az adatbáziskapcsolathoz szükséges adatokat egy titkosított fájlban lettek eltárolva, amely biztosítja az adatok védelmét. Ezt a fájlt is fel kellett tölteni a Streamlit Cloud-ra, hogy az alkalmazás megfelelően hozzáférhessen az adatbázishoz.

4.6.2. Alkalmazás futtatása

A megosztás után bárki számára elérhető az alkalmazás egy böngészőn keresztül. Nem igényel semmilyen telepítést, miután betöltött az oldal egyből lehet használni. Az egyes funkciók végrehajtása több időt vehet igénybe futás közben, mivel az adatbázis műveletek, a gépi tanulási algoritmusok számítása és az API kapcsolatok is időigényes folyamatok. Mivel a

Streamlit Cloud és az adatbázis felhőalapú tárolása ingyenes szolgáltatásokat használ, ezért a műveletek nem a lehető leggyorsabban futnak le.

Az alkalmazás hardvererőforrás-igényét a psutil könyvtár segítségével mértem, amely a CPU- és memóriahasználatot figyelte a teszt során. A tesztelés után megállapítható, hogy 300-600 MB memóriát használ az alkalmazás a különböző funkciók használata közben. A CPU használat a legtöbb esetben elhanyagolható volt, egyedül a bejelentkezés során az API kapcsolat létesítésekor nőtt meg jelentősen. A tesztelés során egy 4 magos processzor 20%-os igénybevételnek volt kitéve. Ez a terhelés jellemzően csak a bejelentkezés idejére volt észlelhető, majd az alkalmazás visszatért az alacsony erőforrás-használathoz. A memóriaszükséglet és a CPU használat tesztelésének az eredményi alapján megállapítható, hogy az alkalmazás hatékonyan képes futni kisebb teljesítményű rendszereken is.

Összefoglalás

A projekt célkitűzéseiben központi szerepet játszott az adattudomány és a gépi tanulás alapfogalmainak elsajátítása, valamint ezen módszerek alkalmazása. A projekt elkészítésekor az elvárás egy olyan predikciós modell készítése volt, amely a kiinduló adatbázis alapján a lehető legmegbízhatóbb eredménnyel tudjon szolgálni. A két feladathoz tartozó tanító adatbázis eltérő méretű és összetételű volt, ezért eltérő megközelítést igényeltek.

A megoldás részét képezte az adatok tisztítása, a hiányzó értékek kezelése, a változók skálázása és kódolása, valamint a legjobb teljesítményű gépi tanulási modellek kiválasztása és finomhangolása. A felsorolt lépések elősegítették, hogy végül sikerüljön találni mindkét feladathoz megfelelő modellt, amely kellően megbízható eredménnyel szolgál. Továbbá a bejelentkezési funkció, illetve az API-n keresztül lekért adatok feldolgozása további bővítéseket és funkcionalitást biztosítottak az alkalmazás számára.

A dolgozat készítése során továbbá ismereteket szereztem a gépi tanulási módszerekről, az adatfeldolgozás és tisztítás folyamatáról, valamint az adattudomány gyakorlati alkalmazásairól. Emellett megismerkedtem az adatok vizualizálásának technikáival, a Python nyelv eszközkészletével, valamint olyan keretrendszerekkel, mint a Streamlit, amely lehetőséget biztosít az interaktív adatelemzési és predikciós modellek bemutatására A választott programozási nyelv és keretrendszerek megfelelő választásnak bizonyultak, mivel ezekkel viszonylag egyszerűen és hatékonyan lehetett megoldani a projekt specifikus feladatait. A fejlesztés során megszerzett tapasztalatok hozzájárultak a különböző statisztikai és adatelemzési megközelítések jobb megértéséhez is.

A jövőbeli fejlesztések között szerepelhet több egészségügyi adat integrálása, valamint a predikciós algoritmusok bővítése és finomítása az eredmények pontosságának további fokozása érdekében. A program másik főbb továbbfejlesztési lehetősége az adatlekérés kiterjesztése más okosóra gyártók szolgáltatásaira, a használt API-n keresztül. Ezen fejlesztések megvalósulásával az alkalmazás még szélesebb körben használhatóvá válna, és pontosabb, személyre szabottabb előrejelzéseket nyújtana a felhasználók számára.

Idegen nyelvű összefoglalás (Summary)

The main goal of the project was to understand the basic concepts of data science and machine learning and to apply these methods. The project was designed to develop a predictive model that could provide the most accurate results from the initial dataset. Due to variations in size and composition, the two training datasets required different approaches for the tasks.

The solution involved cleaning the data, handling missing values, scaling and encoding variables, selecting and fine-tuning the best-performing machine learning models. All these steps were crucial in developing a model for both tasks that delivered sufficiently reliable results. Furthermore, the login functionality and the processing of data retrieved via the API further enhanced the application's features and usability.

During the course of the thesis, I managed to gain knowledge about machine learning methods, data processing and cleaning, as well as the practical applications of data science. I also learned about data visualization techniques, the Python programming language, and frameworks like Streamlit, which enables the creation of interactive data analysis and prediction models. The chosen programming language and frameworks proved to be a good choice, as they allowed for the efficient and relatively straightforward resolution of the project’s specific tasks. Moreover, the experience gained during development also contributed to a better understanding of the different statistical and data analysis approaches.

Future improvements could include integrating additional health data and further expanding and refining the prediction algorithms to enhance the accuracy of the results. Another significant development would be extending data retrieval to include services from other smartwatch vendors via the existing API. If these enhancements are implemented, the application would become even more versatile and capable of providing even more accurate and personalized forecasts to users.

Ábrajegyzék

Irodalomjegyzék

[1] PyPi
Elektronikus forrás: https://pypi.org/

[2] Scikit-learn
Elektronikus forrás: https://scikit-learn.org/stable/

[3] PosgreSQL
Elektronikus forrás: https://www.postgresql.org/docs/

[4] Kaggle
Elektronikus forrás: https://www.kaggle.com/

[5] National Library of Medicine
Elektronikus forrás: https://pmc.ncbi.nlm.nih.gov/articles/PMC2900197/

[6] Kovács Péter: A multikollinearitás vizsgálata lineáris regressziós modellekben
Elektronikus forrás:
https://www.ksh.hu/statszemle_archive/2008/2008_01/2008_01_038.pdf

[7] Machine Learning Mastery
Elektronikus forrás: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

[8] Microsoft Azure
Elektronikus forrás: https://azure.microsoft.com/hu-hu/resources/cloud-computing-dictionary/what-is-machine-learning-platform

[9] IBM
Elektronikus forrás: https://www.ibm.com/topics/logistic-regression

[10] IBM
Elektronikus forrás: https://www.ibm.com/topics/random-forest

[11] DMLC XGBoost
Elektronikus forrás: https://xgboost.readthedocs.io/en/latest/index.html

[12] IBM
Elektronikus forrás: https://www.ibm.com/topics/knn

[13] PennState Eberly Collage of Science
Elektronikus forrás: https://online.stat.psu.edu/stat857/node/211/

[14] ScienceDirect
Elektronikus forrás: https://www.sciencedirect.com/topics/computer-science/model-accuracy

[15] Analytics Vidhya
Elektronikus forrás: https://www.analyticsvidhya.com/articles/precision-and-recall-in-machine-learning/

[16] Medium Towards Data Science
Elektronikus forrás: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Melléklet

Health_Monitoring_FK.pdf

Forráskód: HealthMonitoring-main.zip

Viczián_Dániel_ZV7QI5_20241025_GAMFIT.pdf