Féléves projekt

Python mérnöki alkalmazásai

BMEGEMIBVP3

Készítette: Árvai Péter

Neptun kód: O38B1K

Tartalomjegyzék

Választott feladat

Biebel Botond, Tuba Balázs: *Féléves Projektek BMEGEMIBVP3 (2024/25/1) 1.3.2. Data Science – Középhaladó Nehézségi Szint*

Az adatok forrása

Federico Soriano Palacios: Stroke Prediction Dataset (2020)

https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset?select=healthcare-dataset-stroke-data.csv

A forrásról:

Az Egészségügyi Világszervezet (WHO) szerint a stroke világszerte a 2. vezető halálok, az összes haláleset körülbelül 11%-áért felelős. Ez az adathalmaz arra szolgál, hogy előre jelezze, vajon egy páciens esetében valószínűsíthető-e stroke kialakulása, az olyan bemeneti paraméterek alapján, mint például a nem, az életkor, különböző betegségek jelenléte és a dohányzási szokások. Az adathalmaz minden sora egy páciensről tartalmaz releváns információkat. **[1]**

Attribútumok ismertetése: [1]

- *id:* Egyedi azonosító.
- *gender:* A páciens neme ("Male", "Female", "Other").
- *age:* A páciens életkora.
- *hypertension:* 0, ha a páciensnek nincs magas vérnyomása, 1, ha van.
- *heart_disease:* 0, ha a páciensnek nincs szívbetegsége, 1, ha van.
- *ever_married:* "No", ha a páciens soha nem volt házas, "Yes", ha volt.
- *work_type:* Foglalkozás típusa ("children", "Govt_job", "Never_worked", "Private", "Self-employed").
- *residence_type:* Lakóhely típusa ("Rural", "Urban").
- *avg_glucose_level:* Átlagos vércukorszint.
- *bmi:* Testtömeg-index (BMI).
- *smoking_status:* Dohányzási szokások ("formerly smoked", "never smoked", "smokes", "Unknown").
    - *Megjegyzés*: Az "Unknown" esetében a dohányzási státuszra vonatkozó információ nem elérhető az adott páciens esetében.
- *stroke:* 1, ha a páciensnek volt stroke-ja, 0, ha nem.

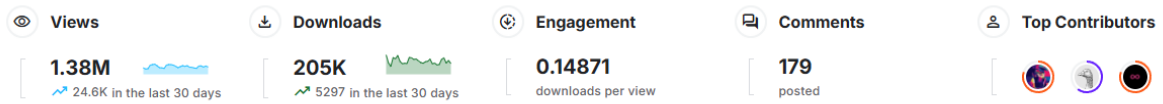
Megjegyzés a szerzőtől: [1]

Az adathalmaz csak oktatási célra használható, és ha kutatás során felhasználásra kerül, az eredeti szerzőt szükséges megjelölni.

A forrás felhasználásáról: [1]

A szerző Stroke Prediction Dataset (healthcare-dataset-stroke-data.csv) nevű adathalmaza ideális választás a választott projekthez, hisz a benne található adatok rendezettek, jól strukturáltak, hibamentesek, az adathalmaz egésze rendkívül jól dokumentált.

Az adathalmaz továbbá nagy népszerűségnek örvend a kaggle.com adatelemzői körében, hiszen jól lehet a segítségével különböző adatelemzési technikákat gyakorolni, prezentálni, sőt akár nem új kódokat és metódusokat kipróbálni.



A projekt célja

A projekt során célom, hogy egy adattudományi tématerülethez kapcsolódó elemzést végezzek el, amely során egy általam kiválasztott adatkészleten alkalmazom az alapvető adatelemzési és -feldolgozási módszereket. Ezen keresztül szeretném elmélyíteni az adattudomány gyakorlati ismereteit, és fejleszteni az elemzési készségeimet. **[2]**

A projekt három nehézségi szint közül választható, és én a középhaladó szintet választottam, amely magában foglalja az alapvető feladatokat, miközben lehetőséget biztosít összetettebb elemzések és technikák alkalmazására is. Ezáltal a projekt során nemcsak az adathalmaz feltérképezésére és az alapvető összefüggések azonosítására fókuszálok, hanem a mélyebb adatelemzési és vizualizációs módszerek alkalmazására is. **[2]**

A feladat során célkitűzéseim a következők:

- Az általam választott adathalmaz szerkezetének és jellemzőinek részletes feltérképezése.
- Az adatok közötti összefüggések és trendek azonosítása, elemzése, valamint ezek interpretálása.
- Az elemzés során alkalmazott módszerek és elért eredmények részletes dokumentálása, különös figyelmet fordítva a szakmai pontosságra és a megállapítások világos megfogalmazására.

A projekthez a **középhaladó szintet** választottam, mert úgy gondolom ez kellő kihívást ad az adattudományban való jártasságom fejlesztésére, de egyben a jelenlegi tudásommal reálisan megvalósítható nehézségi szint.

Feladatok ismertetése

A projekt során az alábbi feladatokat kell elvégezni a kiválasztott adatkészleten: **[2]**

Alapvető feladatok:

1. *Az adatkészlet betöltése és elemzése:*
    - Az adatok betöltése egy megfelelő adattáblába.
    - Az adatok alapvető jellemzőinek vizsgálata, például a méret, a változók, valamint néhány minta rekord megtekintése.
2. *Adatfeltárás:*
    - Az adatok eloszlásának és alapvető statisztikáinak vizsgálata.
    - Átlag, medián, szórás, minimum és maximum értékek elemzése.
    - Hiányzó értékek azonosítása.
3. *Adat előkészítés:*
    - Hiányzó értékek pótlása vagy kezelése.
    - Értelmetlen vagy hibás adatok javítása.
    - Az adathalmaz tisztítása a további elemzésekhez.
4. *Adatvizualizáció:*
    - Legalább három különböző diagram vagy grafikon készítése az adatok megjelenítésére (például oszlopdiagram, kördiagram, hisztogram).
5. *Korreláció elemzés:*
    - A változók közötti kapcsolatok vizsgálata.
    - A legerősebb korrelációk azonosítása és bemutatása.
6. *Hipotézisvizsgálat:*
    - Legalább egy hipotézis megfogalmazása az adatok alapján (előzetes feltételezés).
    - A hipotézis tesztelése és értékelése az adatok elemzése során.
7. *Eredmények értékelése:*
    - Az elvégzett elemzések és azok eredményeinek dokumentálása.

Kezdő feladatok:

8. *Adatlekérdezés:*
    - Egyszerű SQL-lekérdezések az adatkészletből.
9. *Előzetes elemzés:*
    - Az adatkészlet alapvető jellemzőnek vizsgálata, például az értéktartomány és az eloszlás.
10. *Egyszerű adatvizualizáció:*
    - Legalább két különböző típusú diagramot vagy grafikon készítése az adatok vizualizálásához.
11. *Egyszerű statisztikák:*
    - Egyszerű statisztikai mutatók számolása, például átlag és medián.
12. *Alapvető gépi tanulási modell:*
    - Alapvető gépi tanulási modell készítése és teljesítményének kiértékelése.

Középhaladó feladatok:

13. *Haladó gépi tanulási modell:*
    - Gépi tanulási modell létrehozása és finomhangolása.
    - A modell teljesítményének értékelése.
14. *Összetett adatvizualizáció:*
    - Legalább öt különböző diagram vagy grafikon készítése.
    - Az adatok összetett és releváns szempontok szerinti vizualizálása.
15. *Többváltozós statisztika:*
    - A változók közötti összetett statisztikai kapcsolatok elemzése.
16. *Speciális hipotézisvizsgálat:*
    - Különleges hipotézisek vizsgálata az adatok alapján.
    - Az eredmények alapos értékelése.

Alapvető feladatok

1. Az adatkészlet betöltése és elemzése

A kódrészlet betölti az adathalmazt, megjeleníti annak méretét, oszlopainak típusait, egyszerű statisztikai jellemzőit, valamint az első és utolsó 5 rekordját, hogy áttekintést nyújtson az adatok szerkezetéről és tartalmáról.

Input:

```
import pandas as pd

# Az adathalmaz betöltése egy megfelelő adattáblába
data = pd.read_csv("healthcare-dataset-stroke-data.csv")

# Adathalmaz mérete
print("Az adathalmaz mérete (sorok, oszlopok):", data.shape)

# Oszlopok és adattípusok
print("\nOszlopok és adattípusok:")
data.info()

# Első 5 sor
print("\nAz adathalmaz első 5 sora:")
print(data.head())

# Uccsó 5 sor
print("\nAz adathalmaz utolsó 5 sora:")
print(data.tail())
```

Output összegzése:

Adathalmaz mérete: 5110 sor, 12 oszlop.

Oszlopok típusa:

Numerikus: age, avg_glucose_level, bmi.

Bináris: hypertension, heart_disease, stroke.

Kategóriális: gender, ever_married, work_type, Residence_type, smoking_status.

Hiányzó adatok: A bmi oszlopban fordulnak elő hiányzó értékek.

Adatok mintái:

Az első 5 sor stroke-eseteket mutat.

Az utolsó 5 sor stroke nélküli eseteket tartalmaz.

További észrevétel: Az adatstruktúra tiszta és jól szervezett, alkalmas további elemzésre

2. Adatfeltárás

A kódrészlet kiszámítja az adatok alapvető statisztikai jellemzőit (átlag, szórás, medián stb.) és azonosítja a hiányzó értékeket az oszlopokban.

Input:

```
# Alapvető statisztikák
distribution_stats = data.describe().T

# Medián
distribution_stats['median'] = data.median(numeric_only=True)

# Statisztikák
print("Az adatok statisztikai jellemzői:")
print(distribution_stats)

# Hiányzó értékek
missing_values = data.isnull().sum()
print("\nHiányzó értékek oszloponként:")
print(missing_values)
```

Output összegzése:

Életkor (age): Átlag: 43.23 év, medián: 45 év, minimum: 0.08 év, maximum: 82 év.

Átlagos glükózszint (avg_glucose_level):

Átlag: 106.15, medián: 91.89, maximum: 271.74.

Testtömegindex (bmi):

Átlag: 28.89, medián: 28.1, maximum: 97.6.

Bináris változók (pl. hypertension, heart_disease, stroke) értékei jellemzően 0 vagy 1.

Hiányzó értékek:

Csak a bmi oszlopban fordul elő hiányzó adat, összesen 201 érték hiányzik.

A többi oszlop hiánytalan.

3. Adat előkészítés

A kódrészlet betölti az adathalmazt, pótolja a hiányzó BMI értékeket, eltávolítja a gender oszlop Other kategóriáját, törli a duplikátumokat, majd kiírja az adathalmaz tisztított méretét és hiányzó értékeit.

Input:

```
# Hiányzó értékek pótlása (átlaggal a BMI oszlopban)
data['bmi'] = data['bmi'].fillna(data['bmi'].mean())

# Hibás vagy értelmetlen adatok eltávolítása ('gender' oszlopból az 'Other'
kategória)
data = data[data['gender'] != 'Other']

# Duplikált sorok eltávolítása
data = data.drop_duplicates()

# Adathalmaz tisztasága
data_shape_after_cleaning = data.shape
missing_values_after_cleaning = data.isnull().sum()

print("Adathalmaz tisztítása után:")
print("Adathalmaz mérete:", data_shape_after_cleaning)
print("Hiányzó értékek oszloponként:")
print(missing_values_after_cleaning)
```

Output összegzése:

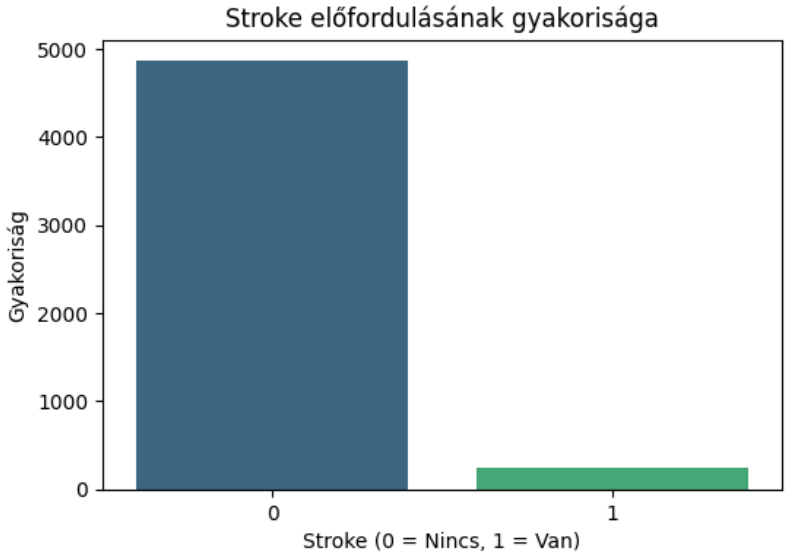
Sikeresen előkészíti. Készen áll a dataset a további elemzésekre vagy modellezésre.

4. Adatvizualizáció:

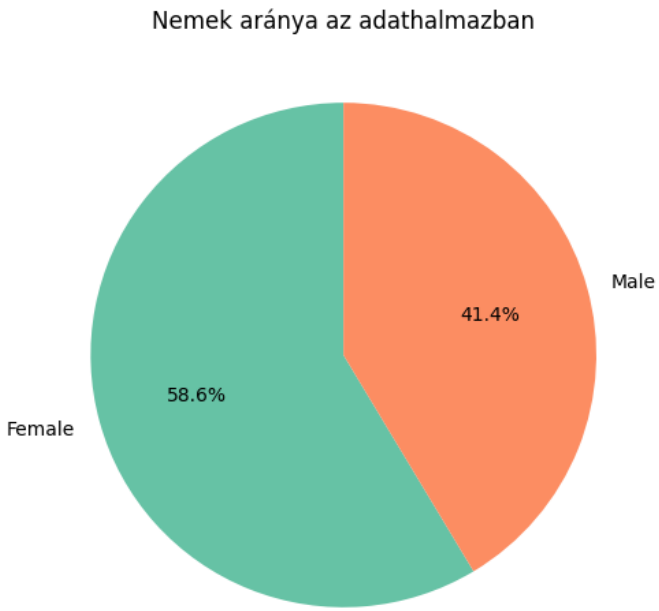
A kódrészlet egy oszlopdiagramot, egy kördiagramot és egy hisztogramot készít.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Oszlopdiagram: A stroke előfordulásának gyakorisága
plt.figure(figsize=(6, 4))
sns.countplot(data=data, x='stroke', palette='viridis')
plt.title('Stroke előfordulásának gyakorisága')
plt.xlabel('Stroke (0 = Nincs, 1 = Van)')
plt.ylabel('Gyakoriság')
plt.show()

# Kördiagram: Nemek aránya az adathalmazban
gender_counts = data['gender'].value_counts()
plt.figure(figsize=(6, 6))
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%',
startangle=90, colors=['#66c2a5', '#fc8d62'])
plt.title('Nemek aránya az adathalmazban')
plt.show()

# Hisztogram: Glükózszint eloszlása
plt.figure(figsize=(8, 5))
sns.histplot(data['avg_glucose_level'], kde=True, color='skyblue', bins=20)
plt.title('Glükózszint eloszlása')
plt.xlabel('Átlagos glükózszint')
plt.ylabel('Gyakoriság')
plt.show()
```

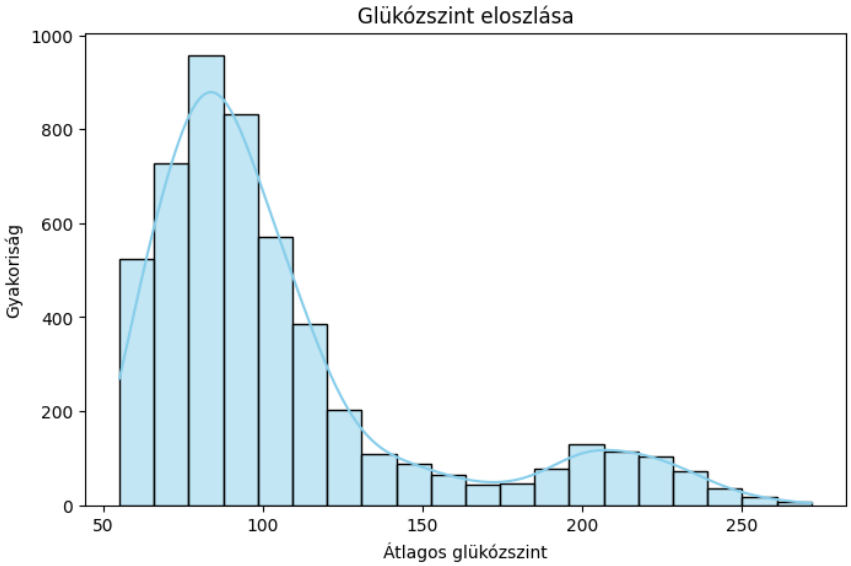
Output:



Oszlopdiagram



Kördiagram



Hisztogram és KDE görbe

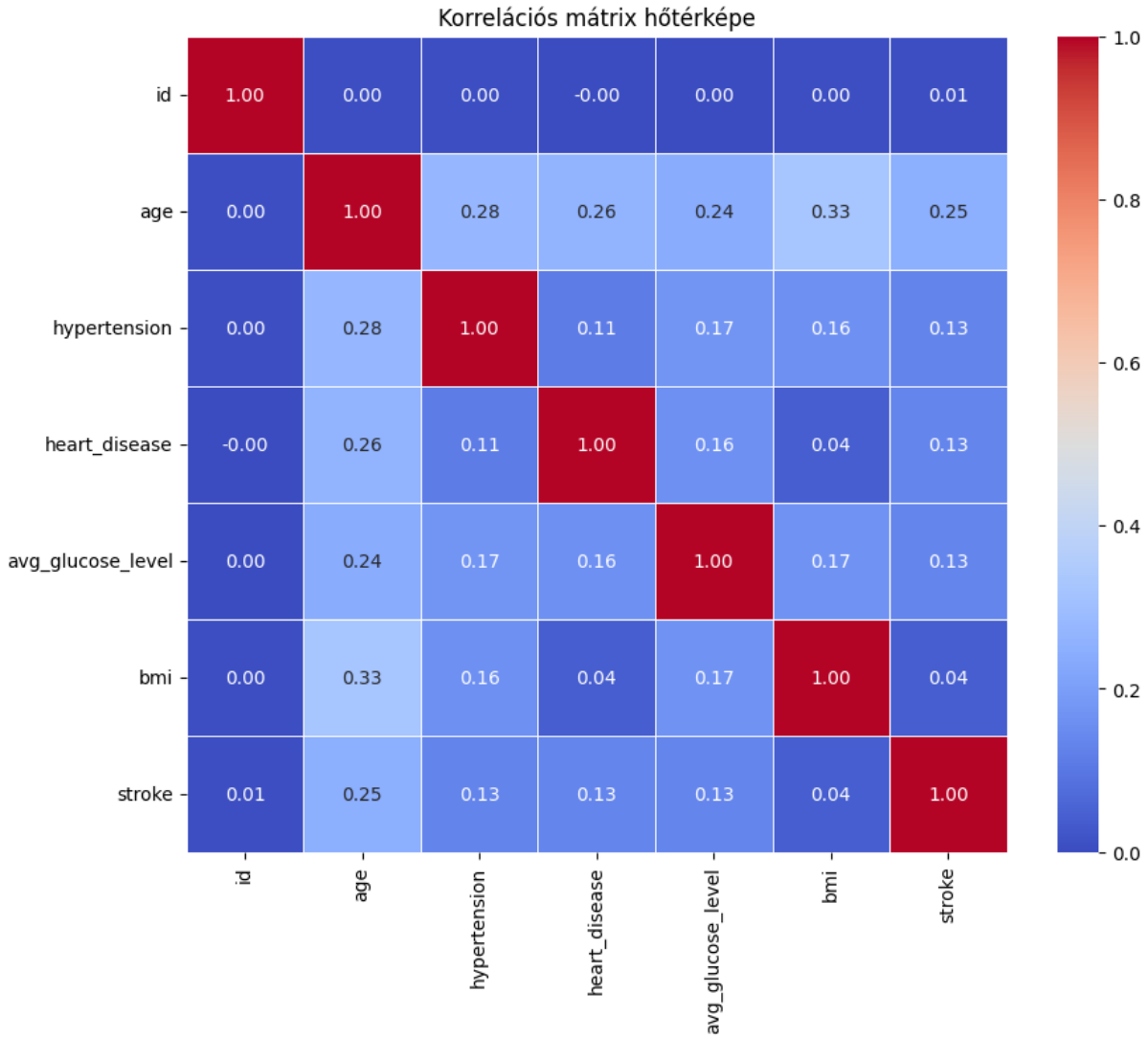
5. Adatvizualizáció:

A kódrészlet a numerikus oszlopokra korlátozva kiszámítja a korrelációs mátrixot, vizualizálja hőtérképen, és azonosítja a legerősebb pozitív és negatív korrelációkat.

Input:

```
# Csak numerikus oszlopokat tartalmazó DataFrame
numerical_data = data.select_dtypes(include=['float64', 'int64'])

# Korrelációs mátrix
correlation_matrix = numerical_data.corr()

# Hőtérkép
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",
linewidths=0.5)
plt.title('Korrelációs mátrix hőtérképe')
plt.show()

# A legerősebb korrelációk azonosítása
strong_correlations =
correlation_matrix.unstack().sort_values(ascending=False).drop_duplicates()
strong_correlations =
strong_correlations[strong_correlations.index.get_level_values(0) !=
strong_correlations.index.get_level_values(1)]

# Legrészletesebb korrelációk kiemelése (küszöbérték legyen: 0.4-nél erősebb)
strong_positive = strong_correlations[strong_correlations > 0.4]
strong_negative = strong_correlations[strong_correlations < -0.4]

print("Erős pozitív korrelációk (> 0.4):")
print(strong_positive)
print("\nErős negatív korrelációk (< -0.4):")
print(strong_negative)
```

Output:



6. Hipotézisvizsgálat:

Azoknál az embereknél, akik hipertóniában szenvednek, nagyobb valószínűséggel fordul elő stroke, mint azoknál, akiknél nincs hipertónia.

Nullhipotézis H_0 **:** Nincs kapcsolat a hipertónia és a stroke előfordulása között.

Alternatív hipotézis H_1 **:** A hipertónia szignifikánsan növeli a stroke előfordulásának valószínűségét.

Az elemzéshez khi-négyzet tesztet használjunk, mivel két kategóriális változó közötti kapcsolatot vizsgálunk.

Input:

```
import scipy.stats as stats

# Hipertónia és stroke gyakorisági táblázat létrehozása
contingency_table = pd.crosstab(data['hypertension'], data['stroke'])

# Khi-négyzet teszt
chi2, p, dof, expected = stats.chi2_contingency(contingency_table)

print("Khi-négyzet érték:", chi2)
print("P-érték:", p)
print("Szabadságfokok:", dof)

# Nullhipotézis jó-e
alpha = 0.05
if p < alpha:
    print("Elutasítjuk a nullhipotézist: van kapcsolat a hipertónia és a
stroke között.")
else:
    print("Nem tudjuk elutasítani a nullhipotézist: nincs szignifikáns
kapcsolat a hipertónia és a stroke között.")
```

Output:

Khi-négyzet érték: 81.57314462043591

P-érték: 1.688936253410575e-19

Szabadságfokok: 1

Elutasítjuk a nullhipotézist: van kapcsolat a hipertónia és a stroke között.

7. Eredmények értékelése:

Az adathalmazzal minden alapvető feladat sikeresen elvégezhető volt.

Kezdő feladatok:

Megjegyzés: Számos feladatot a kezdő feladatok közül megcsináltam az alapvető feladatok között. Ezen feladatoknál a megoldás külön jelölve van.

8. Adatlekérdezés:

Az SQL-lekérdezéseket a Pandas DataFrame-ekkel szimulál.

Egyszerű lekérdezések pl:

SELECT: Az első 5 rekord megtekintése.

WHERE: Hipertóniás betegek szűrése.

COUNT: A stroke-esetek számának meghatározása.

GROUP BY: Átlagos glükózszint kiszámítása nemek szerint.

ORDER BY: Legmagasabb BMI-vel rendelkező betegek listázása.

Input:

```
import pandasql as ps

data = pd.read_csv("healthcare-dataset-stroke-data.csv")
# SELECT
query1 = "SELECT * FROM data LIMIT 5;"
result1 = ps.sqldf(query1, locals())
print("Az első 5 rekord:\n", result1)

# WHERE
query2 = "SELECT * FROM data WHERE hypertension = 1;"
result2 = ps.sqldf(query2, locals())
print("\nHipertóniás betegek:\n", result2.head())

# COUNT
query3 = "SELECT COUNT(*) AS stroke_cases FROM data WHERE stroke = 1;"
result3 = ps.sqldf(query3, locals())
print("\nStroke-esetek száma:\n", result3)

# GROUP BY
query4 = "SELECT gender, AVG(avg_glucose_level) AS avg_glucose FROM data GROUP
BY gender;"
result4 = ps.sqldf(query4, locals())
print("\nÁtlagos glükózszint nemek szerint:\n", result4)

# ORDER BY
query5 = "SELECT * FROM data ORDER BY bmi DESC LIMIT 5;"
result5 = ps.sqldf(query5, locals())
print("\nLegmagasabb BMI-vel rendelkező betegek:\n", result5)
```

Output összegzése:

Az első 5 rekord:

Az első öt beteg életkora 49 és 80 év között van, többségük stroke-ot szenvedett el, és magas glükózszintjük van.

Hipertóniás betegek:

A hipertóniás betegek életkora jellemzően magas (50 év felett). A legtöbben vidéki lakosok és nem dohányoznak.

Stroke-esetek száma:

Az adathalmazban 249 beteg szerepel stroke-diagnózissal.

Átlagos glükózszint nemek szerint:

A férfiak átlaga magasabb (109.09), mint a nőké (104.06). Az Other kategóriába tartozó egyének átlaga kiemelkedően magas (143.33).

Legmagasabb BMI-vel rendelkező betegek:

A legmagasabb BMI-vel rendelkező egyének fiatalabbak, többségük hipertóniás, és a stroke előfordulása alacsony ebben a csoportban.

9. Előzetes elemzés 2.0:

A kódrészlet:

- *betölti az adatokat és kiválasztja a numerikus oszlopokat*
- *meghatározza a statisztikai jellemzőket*
- *hiszotgramokat és KDE görbéket készít*
- *IQR módszerrel vizsgálja a kiugró értékeket*

Input:

```
numerical_columns = data.select_dtypes(include=['float64', 'int64'])

# Statisztikai jellemzők, értéktartományok
range_info = numerical_columns.describe().T[['min', 'max', 'mean', 'std']]
print("Numerikus oszlopok statisztikai jellemzői (értéktartomány, átlag,
szórás):")
print(range_info)

# Hisztogramok és KDE görbék
for column in numerical_columns.columns:
    plt.figure(figsize=(8, 5))
    sns.histplot(numerical_columns[column], kde=True, bins=30, color='blue',
alpha=0.6)
    plt.title(f'{column} eloszlása és KDE görbéje')
    plt.xlabel(column)
    plt.ylabel('Gyakoriság')
    plt.show()

# IQR módszer
outliers = pd.DataFrame()

for column in numerical_columns.columns:
    q1 = numerical_columns[column].quantile(0.25)
    q3 = numerical_columns[column].quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    outliers[column] = ~numerical_columns[column].between(lower_bound,
upper_bound)

print("\nKiugró értékeket tartalmazó sorok száma (oszloponként):")
print(outliers.sum())

total_outliers = outliers.any(axis=1).sum()
print("\nTeljes adathalmazban kiugró értékeket tartalmazó sorok száma:",
total_outliers)
```

Output:

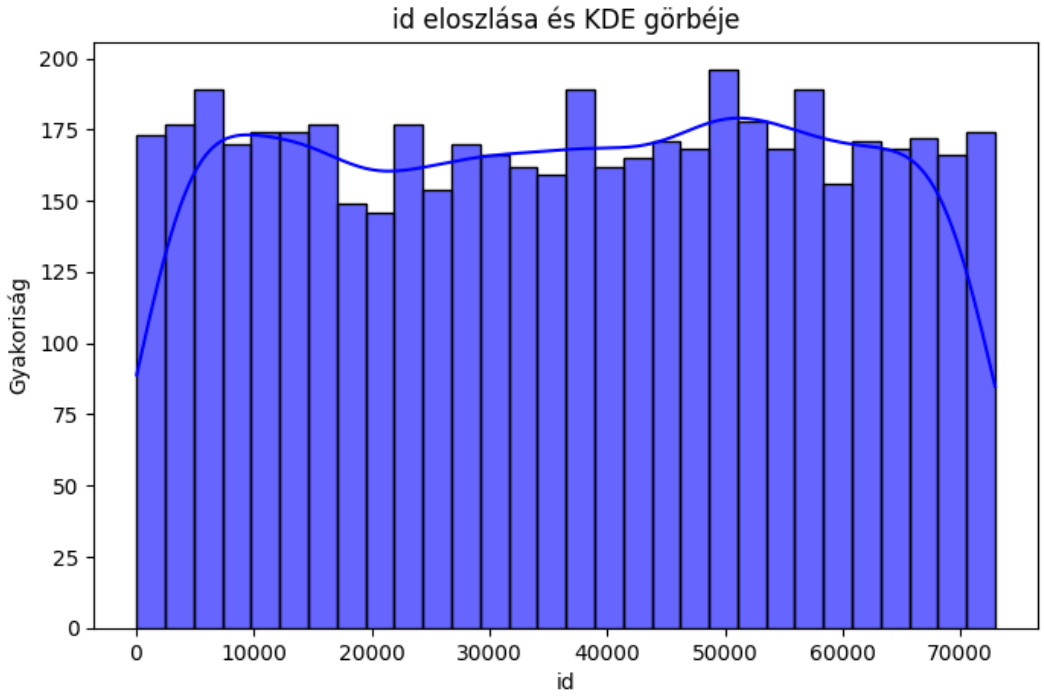
Életkor: 0.08 és 82 év között változik, átlagosan 43 év.

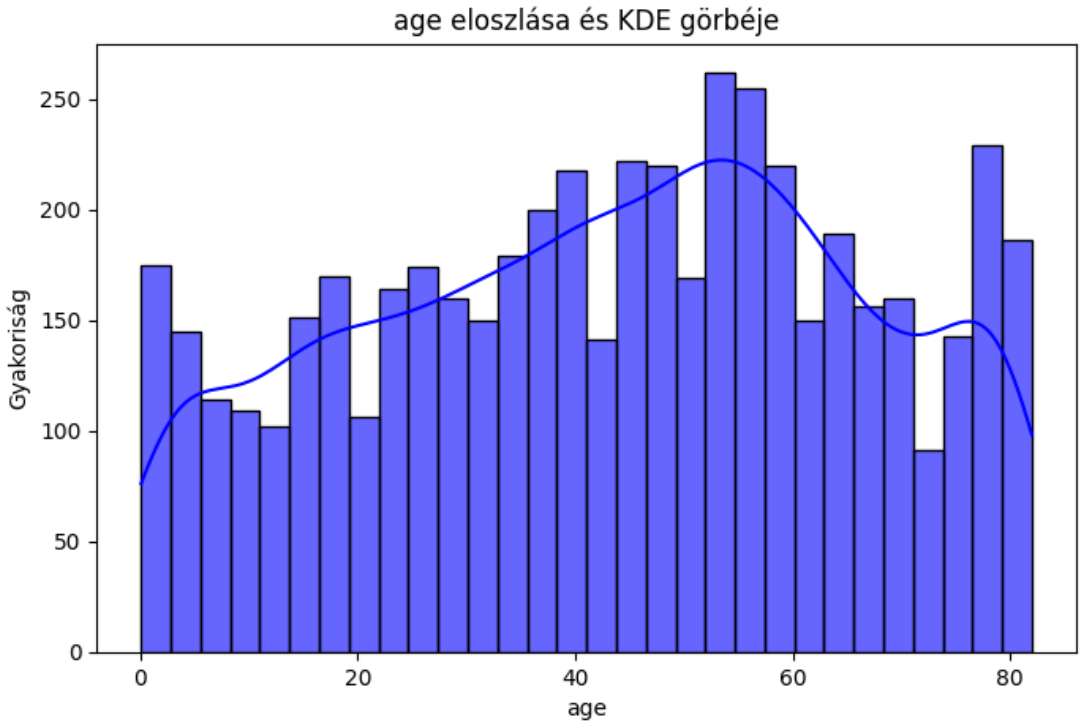
Átlagos glükózszint: 55.12 és 271.74 között, átlaga 106.15.

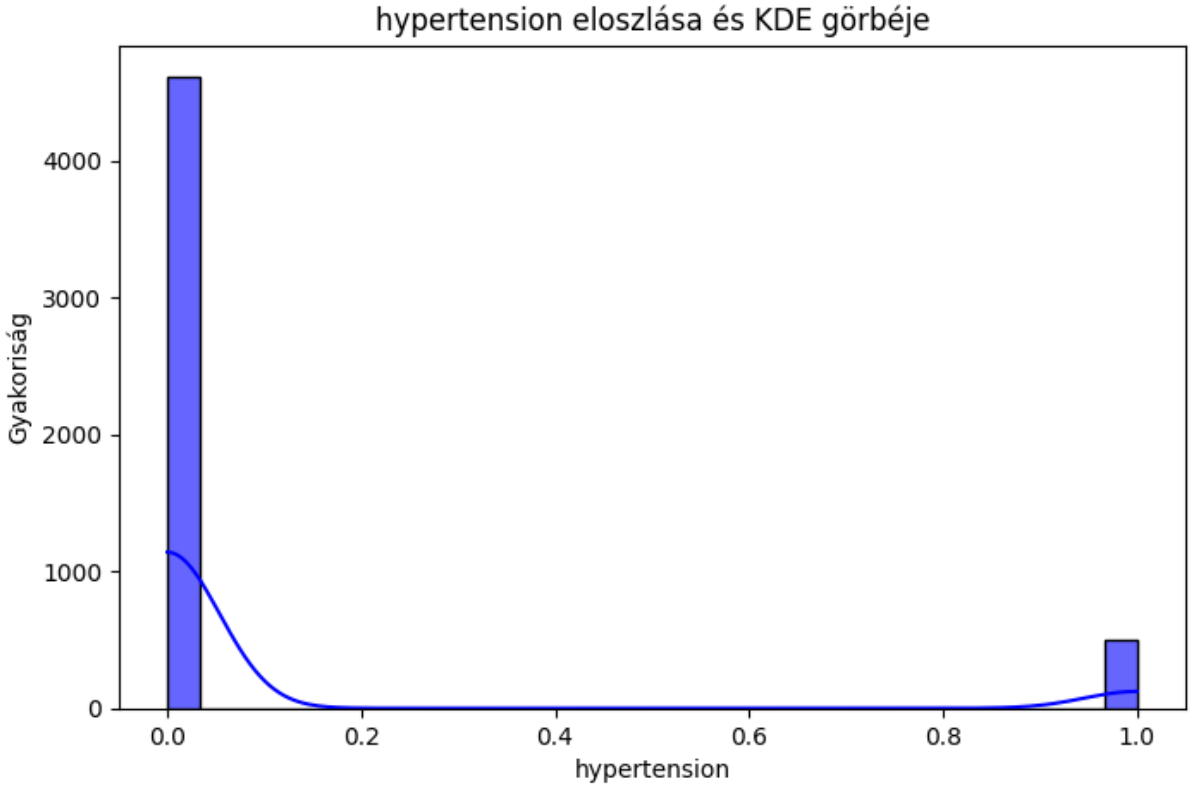
BMI: Értékei 10.30 és 97.60 között változnak, átlaga 28.89.

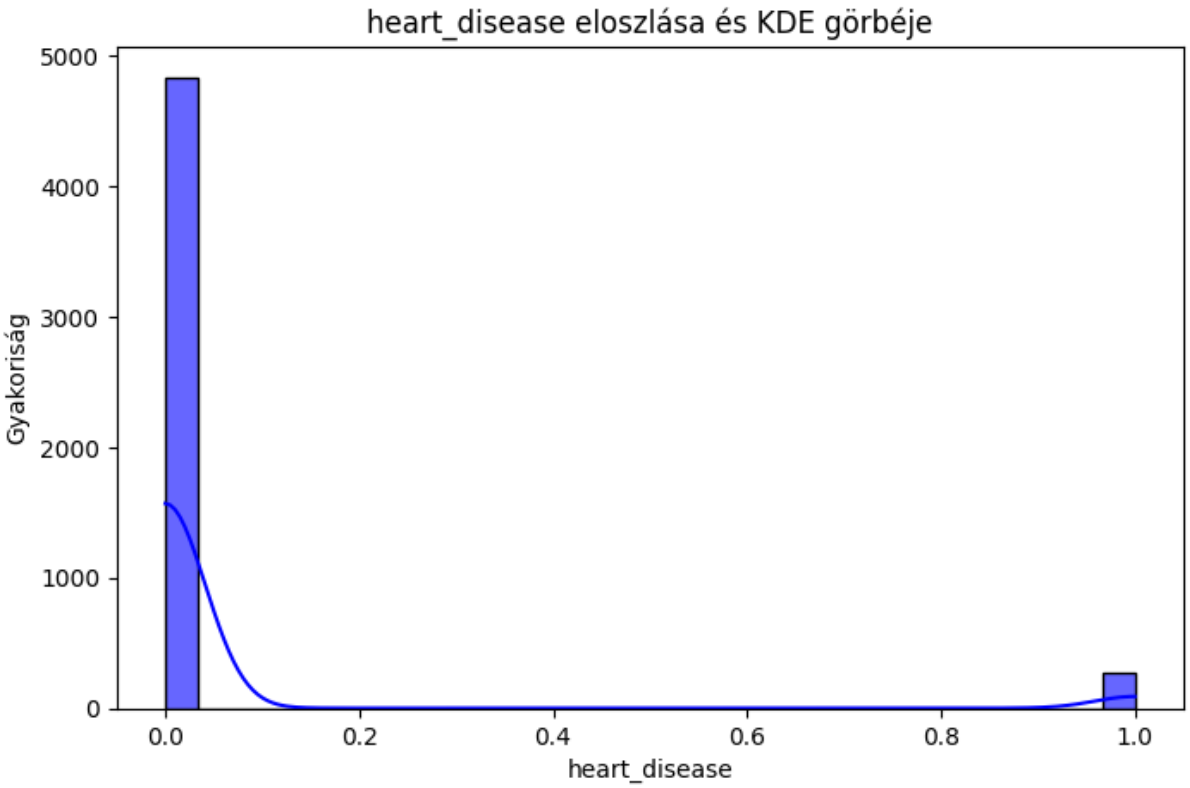
Hipertónia és szívbetegség: Bináris változók (0 vagy 1), ritkán fordulnak elő (átlaguk < 0.1).

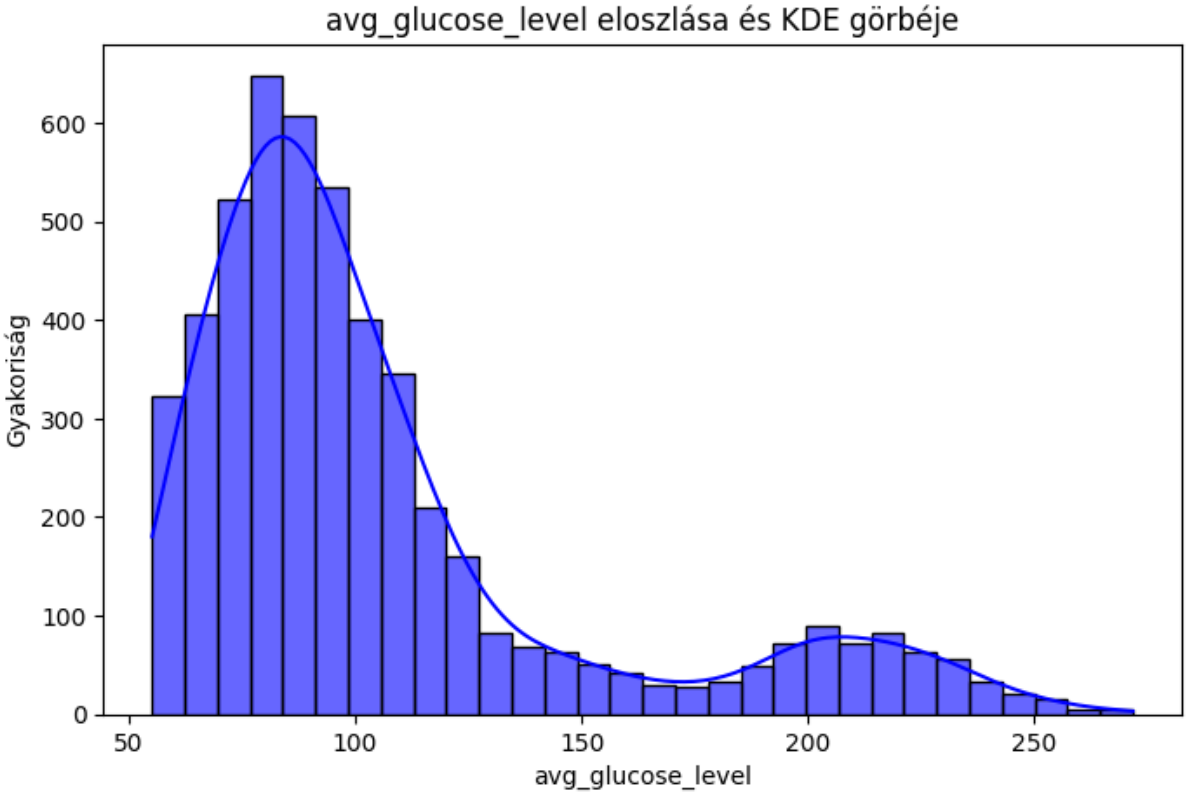
Stroke: Ritka esemény, átlaga 0.0487, ami a stroke előfordulásának alacsony gyakoriságára utal.



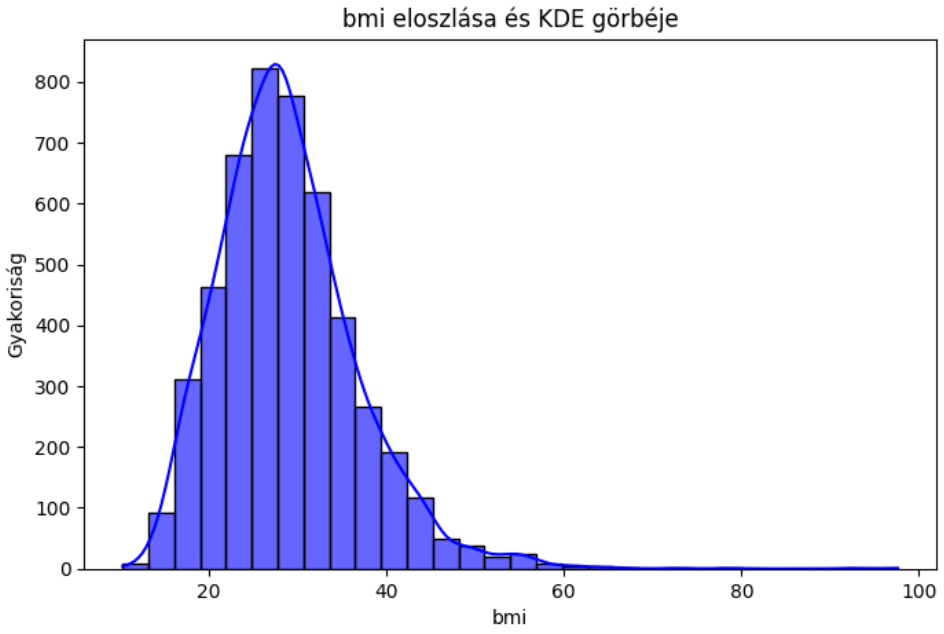

age eloszlása és KDE görbéje
Gyakoriság
age
0
50
100
150
200
250
0
20
40
60
80

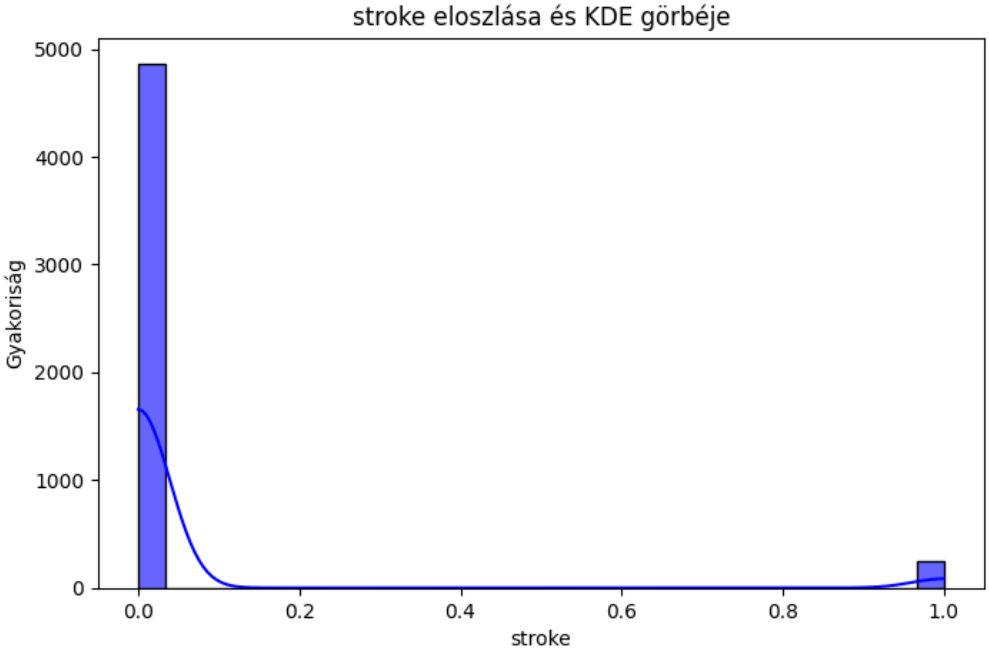

hypertension eloszlása és KDE görbéje
Gyakoriság
hypertension
0
1000
2000
3000
4000
0.0
0.2
0.4
0.6
0.8
1.0





(ez már volt 3. feladatban is!)





<u>Legtöbb kiugró érték:</u>

Átlagos glükózszint: 627 kiugró érték.

Hipertónia: 498 kiugró érték.

<u>Egyéb kiugrók:</u>

BMI: 311 kiugró.

Szívbetegség: 276 kiugró.

Stroke: 249 kiugró.

Összesen: Az adathalmazban 1354 sor tartalmaz kiugró értéket.

10. Egyszerű adatvizualizáció:

Lásd: 2., 9. feladat

11. Egyszerű statisztikák:

Lásd: 2. feladat

12. Alapvető gépi tanulási modell:

A feladatot logisztikus regressziós modell segítségével oldottam meg. Az adathalmazt először előkészítettem (ennek egy része már a 3. feladatban megtörtént): a célváltozót elkülönítettük, a hiányzó értékeket pótoltuk, majd a tanuló- és teszthalmazokra osztottuk. Ezután a modellt betanítottuk a tanulóhalmaz adataival, és kiértékeltük a teszthalmazon a pontosság, osztályozási jelentés és konfúziós mátrix alapján.

Input:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix


# Adatok előkészítése a gépi tanulási modellhez
# Célváltozó (target) és bemeneti változók (features) szétválasztása
X = data.drop(columns=['stroke', 'id', 'gender', 'smoking_status',
'work_type', 'Residence_type', 'ever_married'])
y = data['stroke']

# Hiányzó értékek
X.fillna(X.mean(), inplace=True)

# Adatok szétosztása tanuló és teszt adathalmazra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Log regresszió
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Modell teljesítménye
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print("Modell pontossága (accuracy):", accuracy)
print("\nOsztályozási jelentés (classification report):")
print(classification_report(y_test, y_pred))
print("\nKonfúziós mátrix (confusion matrix):")
print(confusion_matrix(y_test, y_pred))
```

Kiértékelés:

Pontosság (Accuracy):

A modell pontossága 93.93%, ami azt jelzi, hogy az esetek nagy részét helyesen osztályozta.

Problémás tényezők:

Recall és Precision az 1-es osztályra (Stroke):

A modell nem talált egyetlen stroke esetet sem (0 találat a pozitív osztályra).

Ez súlyos egyensúlyhiányra utal az adatokban (az osztályok közötti eloszlás nem egyenletes).

A konfúziós mátrix szerint minden predikció 0-s osztályba esett, az 1-es osztályt figyelmen kívül hagyta.

Átlagok:

A Macro Avg alacsony (0.48), mivel az 1-es osztályt nem találta meg.

A Weighted Avg magasabb (0.91), mert az adatok nagy része a 0-s osztályba tartozik.

Tehát: A modell jól teljesít az általános pontosság szempontjából, de az osztályok egyensúlyhiánya miatt nem képes a stroke-os esetek azonosítására. Érdemes az adatok kiegyensúlyozására stratégiát alkalmazni, például az alábbi módszereket:

Adatosztályok kiegyensúlyozása: SMOTE (Synthetic Minority Oversampling Technique).

Lásd 13. feladat

Osztály súlyozása: Az osztály-súlyok megadása a logisztikus regressziónál.

Középhaladó feladatok:

Megjegyzés: Számos feladatot a középhaladó feladatok közül megcsináltam az alapvető és a kezdő feladatok között. Ezen feladatoknál a megoldás külön jelölve van.

13. Haladó gépi tanulási modell:

A kódrészlet egy Random Forest gépi tanulási modellt hoz létre a stroke előrejelzésére. Az adatok kiegyensúlyozására SMOTE-ot használ, majd a modellt GridSearchCV segítségével finomhangolja, tehát lényegében az 12. feladatot ”javítjuk fel”. Kiértékeli a modellt pontosság, osztályozási jelentés, ROC AUC és konfúziós mátrix alapján, valamint megjeleníti a ROC-görbét.

Input: (a következő oldalon)

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import accuracy_score, roc_auc_score,
classification_report, confusion_matrix
from imblearn.over_sampling import SMOTE
import numpy as np
import matplotlib.pyplot as plt

X = data.drop(columns=['stroke', 'id', 'gender', 'smoking_status',
'work_type', 'Residence_type', 'ever_married'])
y = data['stroke']
X.fillna(X.mean(), inplace=True)

#Többszörös split
n_iterations = 10
accuracies = []
roc_aucs = []

for i in range(n_iterations):
    #Train-test split stratifikáltan
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
stratify=y, random_state=i)

    #SMOTE csak a tanuló adathalmazon
    smote = SMOTE(random_state=42)
    X_train_resampled, y_train_resampled = smote.fit_resample(X_train,
y_train)

    #Finomhangolás
    param_grid = {
        'n_estimators': [100, 200],
        'max_depth': [2, 5, None],
        'min_samples_split': [2, 5],
        'min_samples_leaf': [1, 2]
    }

    rf_model = RandomForestClassifier(random_state=42)
    grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid,
cv=3, scoring='roc_auc', verbose=0)
    grid_search.fit(X_train_resampled, y_train_resampled)

    #Legjobb modell kiértékelése
    best_model = grid_search.best_estimator_
    y_pred = best_model.predict(X_test)
    y_prob = best_model.predict_proba(X_test)[:, 1]

    accuracies.append(accuracy_score(y_test, y_pred))
    roc_aucs.append(roc_auc_score(y_test, y_prob))
```

```
mean_accuracy = np.mean(accuracies)
std_accuracy = np.std(accuracies)
mean_roc_auc = np.mean(roc_aucs)
std_roc_auc = np.std(roc_aucs)

print(f"Modell pontossága (accuracy): {mean_accuracy:.2f} ±
{std_accuracy:.2f}")
print(f"ROC AUC: {mean_roc_auc:.2f} ± {std_roc_auc:.2f}")

#Konfúziós mátrix és ROC görbe
conf_matrix = confusion_matrix(y_test, y_pred)
print("\nKonfúziós mátrix:")
print(conf_matrix)

fpr, tpr, thresholds = roc_curve(y_test, y_prob)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (AUC =
{mean_roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='red', linestyle='--', label='Random Guessing')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.grid()
plt.show()
```

Output:

Modell pontossága (accuracy): 0.87 ± 0.01

ROC AUC: 0.78 ± 0.02

Konfúziós mátrix:

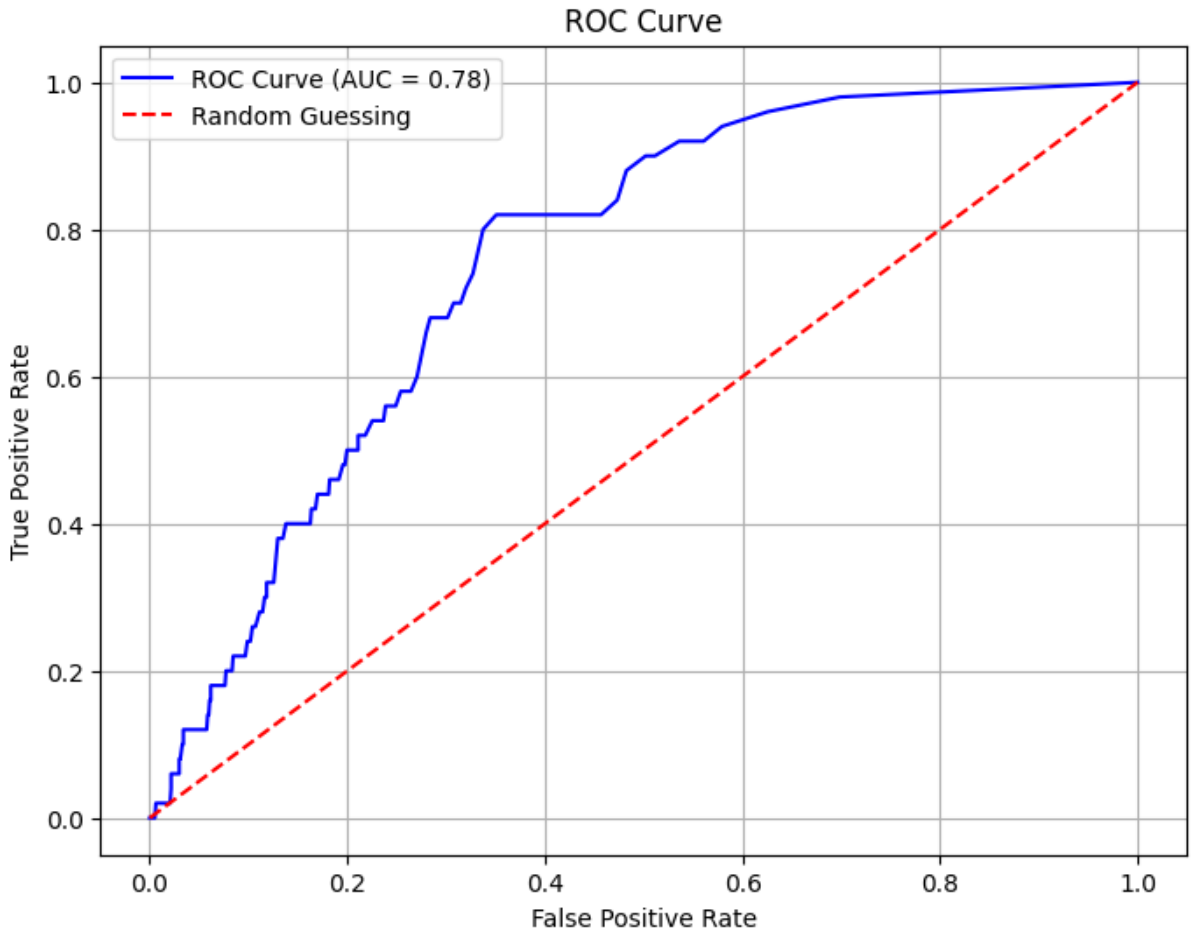
[[884 88]

[39 11]]

A modell pontossága megfelelő, de a stroke-ot szenvedett egyének azonosítása (True Positives) még mindig kihívás.

A magas False Positives és False Negatives számok arra utalnak, hogy a modell további finomhangolást igényel, különösen az egyensúlyhiányos osztályok kezelése terén.

A ROC AUC érték azt mutatja, hogy a modell bizonyos mértékig képes megkülönböztetni a stroke-os eseteket, de nem elég robusztus.



14. Összetett adatvizualizáció:

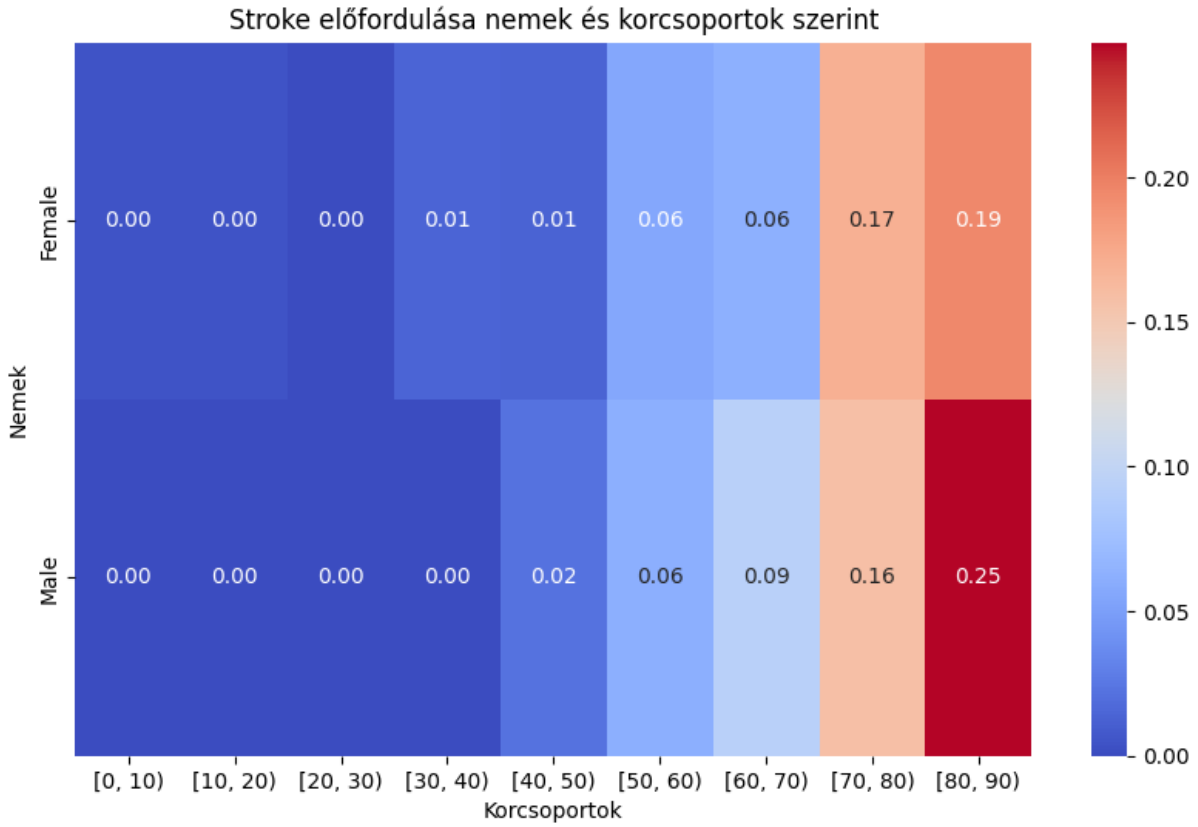
Még több haladóbb, izgalmasabb adatvizualizáció.

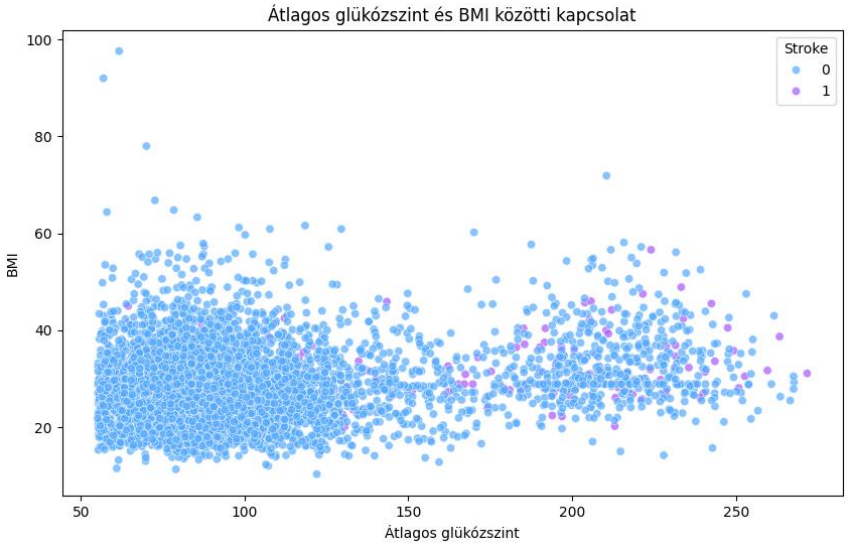
Input:

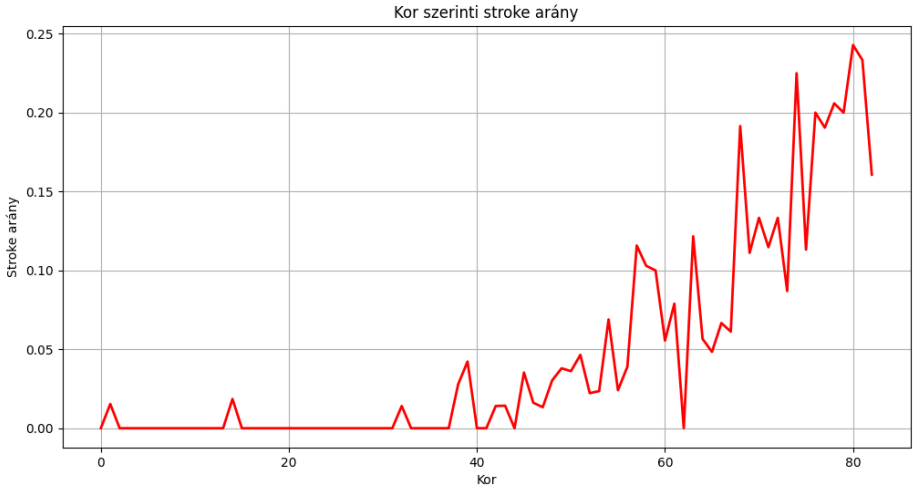
```
# Hőtérkép
data_cleaned['age'] = data_cleaned['age'].round().astype(int)

# Korcsoport-határok 10-esek
bins = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]

# Stroke
plt.figure(figsize=(10, 6))
age_gender_pivot = data_cleaned.pivot_table(
    values='stroke',
    index='gender',
    columns=pd.cut(data_cleaned['age'], bins=bins, right=False),
    aggfunc='mean'
)
sns.heatmap(age_gender_pivot, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Stroke előfordulása nemek és korcsoportok szerint')
```

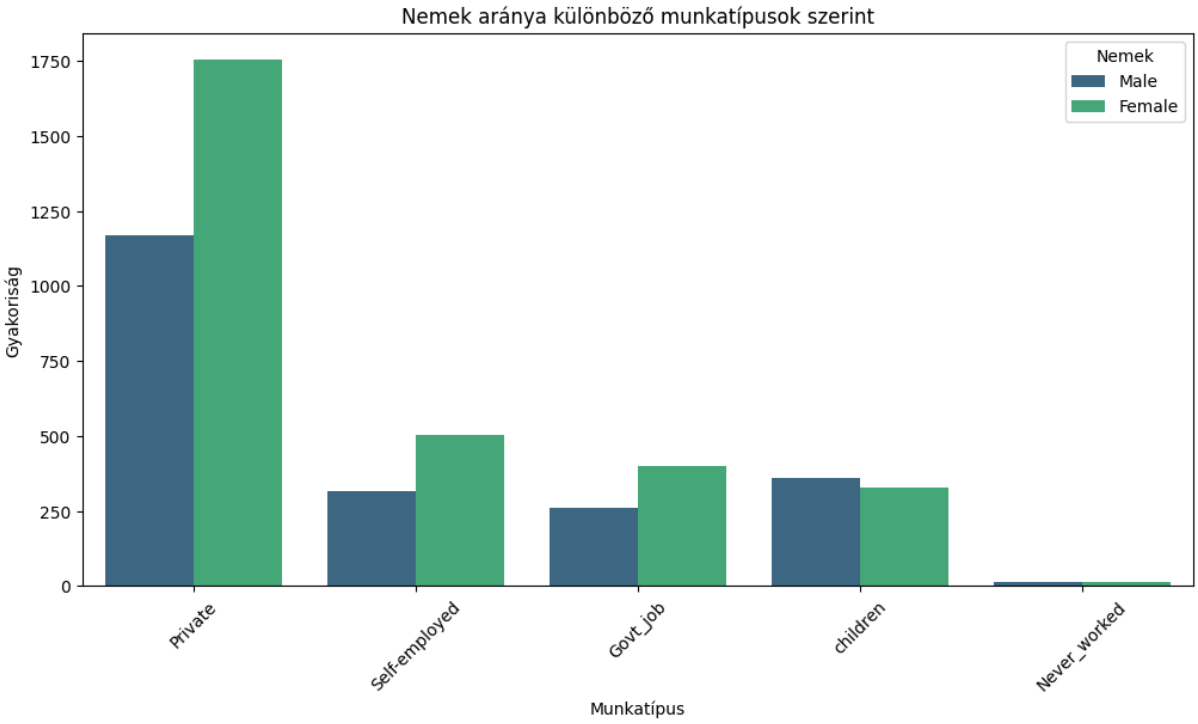
```
plt.xlabel('Korcsoportok')
plt.ylabel('Nemek')
plt.show()
# Szórásdiagram
plt.figure(figsize=(10, 6))
sns.scatterplot(data=data_cleaned, x='avg_glucose_level', y='bmi',
hue='stroke', alpha=0.7, palette='cool')
plt.title('Átlagos glükózszint és BMI közötti kapcsolat')
plt.xlabel('Átlagos glükózszint')
plt.ylabel('BMI')
plt.legend(title='Stroke')
plt.show()

# Vonaldiagram
age_stroke_ratio = data_cleaned.groupby('age')['stroke'].mean()
plt.figure(figsize=(12, 6))
plt.plot(age_stroke_ratio, color='red', linewidth=2)
plt.title('Kor szerinti stroke arány')
plt.xlabel('Kor')
plt.ylabel('Stroke arány')
plt.grid()
plt.show()

# Csoportosított oszlopdiagram
plt.figure(figsize=(12, 6))
sns.countplot(data=data_cleaned, x='work_type', hue='gender',
palette='viridis')
plt.title('Nemek aránya különböző munkatípusok szerint')
plt.xlabel('Munkatípus')
plt.ylabel('Gyakoriság')
plt.legend(title='Nemek')
plt.xticks(rotation=45)
plt.show()

# Interaktív bubble chart
fig = px.scatter(
    data_cleaned,
    x='avg_glucose_level',
    y='bmi',
    size='age',
    color='stroke',
    hover_data=['gender', 'work_type'],
    title='Interaktív Glükózszint és BMI vizsgálat korcsoportokkal',
    labels={'avg_glucose_level': 'Átlagos glükózszint', 'bmi': 'BMI'},
)
fig.update_traces(marker=dict(opacity=0.6, line=dict(width=1,
color='DarkSlateGrey')))
fig.show()
```

Output:


Stroke előfordulása nemek és korcsoportok szerint
Nemek
Female
Male
0.00 0.00 0.00 0.01 0.01 0.06 0.06 0.17 0.19
0.00 0.00 0.00 0.00 0.02 0.06 0.09 0.16 0.25
[0, 10) [10, 20) [20, 30) [30, 40) [40, 50) [50, 60) [60, 70) [70, 80) [80, 90)
Korcsoportok
0.20
0.15
0.10
0.05
0.00


Átlagos glükózszint és BMI közötti kapcsolat
Stroke
0
1
BMI
100
80
60
40
20
50
100
150
200
250
Átlagos glükózszint





15. Többváltozós statisztika:

Minden amit műgazdból tanultunk (amiből remélem nem húznak meg lol)

Korrelációs mátrixok (Pearson és Spearman):

Az első két hőtérkép a numerikus változók közötti lineáris és rangsorrendi kapcsolatokat mutatja. Erősebb korreláció esetén a színek intenzívebbek.

Pearson-korreláció stroke és egyes változók között:

Az értékek megmutatják, hogy az egyes változók (pl. életkor, glükózszint) mennyire kapcsolódnak a stroke-hoz.

Főkomponens-analízis (PCA):

A PCA két dimenzióban ábrázolja a változók közötti kapcsolatokat. A pontok színe a stroke meglétét jelzi.

Glükózszint és BMI kapcsolata:

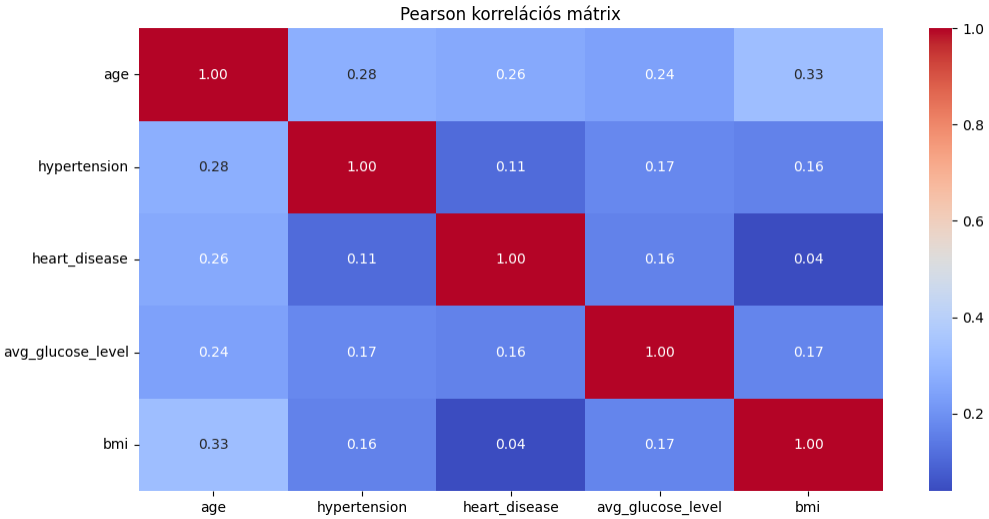
Az utolsó grafikon a glükózszint és a BMI közötti lineáris kapcsolatot szemlélteti, külön kiemelve a stroke meglétével vagy hiányával.

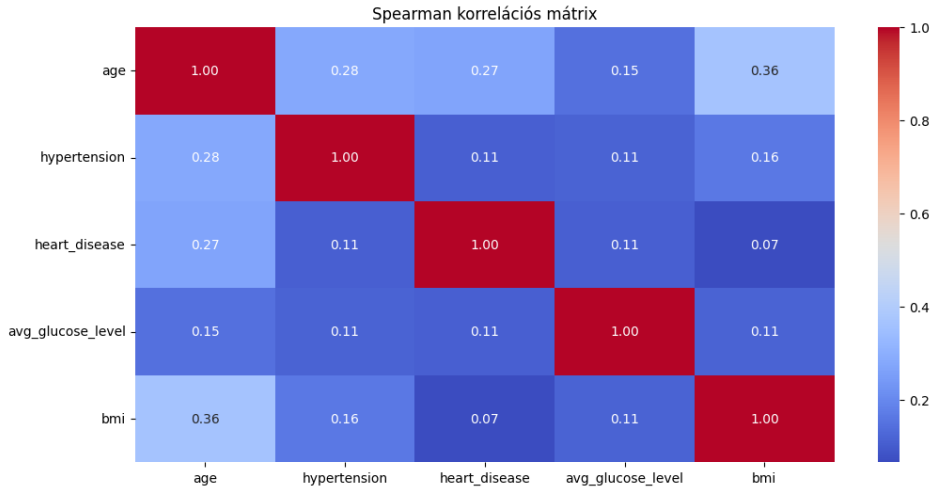
Input:

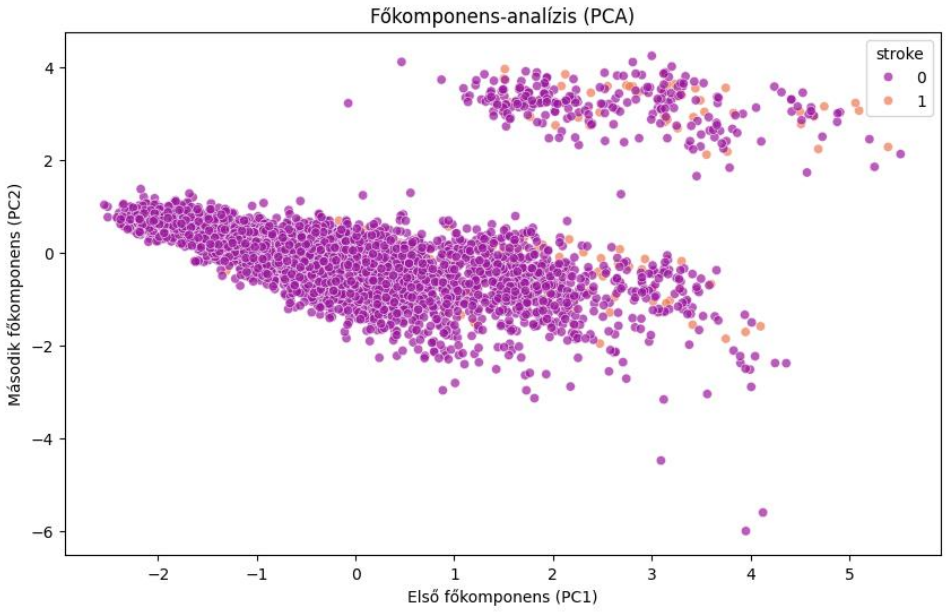
```
numeric_cols = ['age', 'hypertension', 'heart_disease', 'avg_glucose_level',
'bmi']

# Pearson Spearman
pearson_corr = data_cleaned[numeric_cols].corr(method='pearson')
spearman_corr = data_cleaned[numeric_cols].corr(method='spearman')

# Korrelációs hőtérkép
plt.figure(figsize=(12, 6))
sns.heatmap(pearson_corr, annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Pearson korrelációs mátrix")
plt.show()

plt.figure(figsize=(12, 6))
sns.heatmap(spearman_corr, annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Spearman korrelációs mátrix")
plt.show()

# Lineáris összefüggések
for col in numeric_cols:
    if col != 'stroke':
        corr, _ = pearsonr(data_cleaned['stroke'], data_cleaned[col])
        print(f"Pearson korreláció stroke és {col} között: {corr:.2f}")

# Főkomponens-analízis (PCA) az adatok dimenziójának csökkentésére
scaler = StandardScaler()
X_scaled = scaler.fit_transform(data_cleaned[numeric_cols])

pca = PCA(n_components=2)
pca_result = pca.fit_transform(X_scaled)

# Főkomponensek vizualizálása
pca_df = pd.DataFrame(data=pca_result, columns=['PC1', 'PC2'])
pca_df['stroke'] = data_cleaned['stroke'].values

plt.figure(figsize=(10, 6))
sns.scatterplot(data=pca_df, x='PC1', y='PC2', hue='stroke', palette='plasma',
alpha=0.7)
plt.title('Főkomponens-analízis (PCA)')
plt.xlabel('Első főkomponens (PC1)')
```

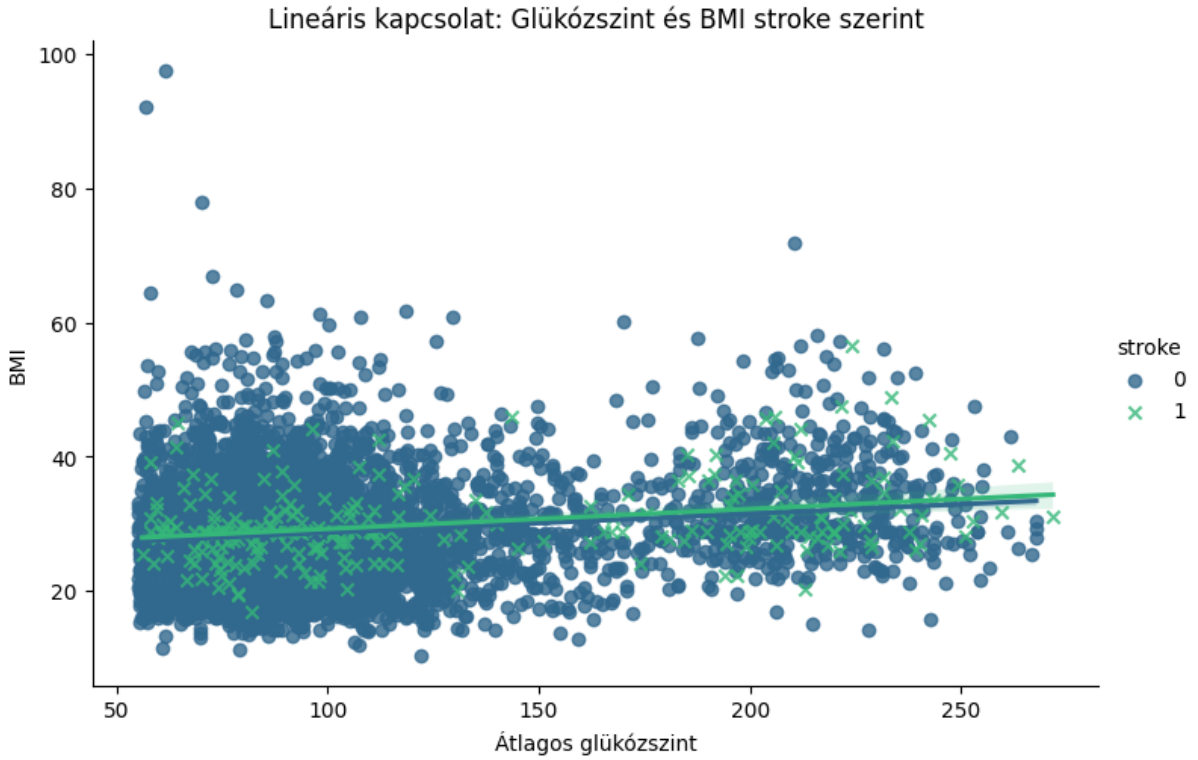
```
plt.ylabel('Második főkomponens (PC2)')
plt.show()

# Glükózszint és BMI interakció
sns.lmplot(
    data=data_cleaned,
    x='avg_glucose_level',
    y='bmi',
    hue='stroke',
    aspect=1.5,
    markers=['o', 'x'],
    palette='viridis'
)
plt.title('Lineáris kapcsolat: Glükózszint és BMI stroke szerint')
plt.xlabel('Átlagos glükózszint')
plt.ylabel('BMI')
plt.show()
```

Output:


Pearson korrelációs mátrix
age
hypertension
heart_disease
avg_glucose_level
bmi
1.00 0.28 0.26 0.24 0.33
0.28 1.00 0.11 0.17 0.16
0.26 0.11 1.00 0.16 0.04
0.24 0.17 0.16 1.00 0.17
0.33 0.16 0.04 0.17 1.00
1.0
0.8
0.6
0.4
0.2


Spearman korrelációs mátrix
age
hypertension
heart_disease
avg_glucose_level
bmi
1.00 0.28 0.27 0.15 0.36
0.28 1.00 0.11 0.11 0.16
0.27 0.11 1.00 0.11 0.07
0.15 0.11 0.11 1.00 0.11
0.36 0.16 0.07 0.11 1.00
1.0
0.8
0.6
0.4
0.2


Főkomponens-analízis (PCA)
stroke
0
1
Második főkomponens (PC2)
Első főkomponens (PC1)
4
2
0
−2
−4
−6
−2
−1
0
1
2
3
4
5



16. Speciális hipotézisvizsgálat:

Van szignifikáns különbség a stroke-ot elszenvedők és a stroke-ot nem elszenvedők életkora között.

Nullhipotézis (H_0): Az életkor nem tér el szignifikánsan a stroke-csoportok között.

Alternatív hipotézis (H_1): Az életkor szignifikánsan eltér a stroke-ot elszenvedők és a nem elszenvedők között.

Nullhipotézis H_0 **:** Az életkor nem tér el szignifikánsan a stroke-csoportok között.

Alternatív hipotézis H_1 **:** Az életkor szignifikánsan eltér a stroke-ot elszenvedők és a nem elszenvedők között.

Az elemzéshez T-próbát használjunk.

Input:

```
from scipy.stats import ttest_ind

stroke_group = data_cleaned[data_cleaned['stroke'] == 1]['age']
non_stroke_group = data_cleaned[data_cleaned['stroke'] == 0]['age']

# T-próba
t_stat, p_value = ttest_ind(stroke_group, non_stroke_group, equal_var=False)

print("T-próba eredményei:")
print(f"T-statisztika: {t_stat:.2f}")
print(f"P-érték: {p_value:.4f}")
```

```
# Nullhipotézis jó-e
if p_value < 0.05:
    print("Elutasítjuk a nullhipotézist: Az életkor szignifikánsan eltér a
stroke-ot elszenvedők és a nem elszenvedők között.")
else:
    print("Nem tudjuk elutasítani a nullhipotézist: Az életkor nem tér el
szignifikánsan a stroke-csoportok között.")
```

Output:

T-statisztika: 29.67

P-érték: 0.0000

Elutasítjuk a nullhipotézist: Az életkor szignifikánsan eltér a stroke-ot elszenvedők és a nem elszenvedők között.

Források és hivatkozások

[1] Biebel Botond, Tuba Balázs: *Féléves Projektek BMEGEMIBVP3 (2024/25/1) 1.3.2. Data Science – Középhaladó Nehézségi Szint* [online] (utoljára megnyitva és használva: 2024.12.03.)
feleves_projekt.pdf

[2] Federico Soriano Palacios: *Stroke Prediction Dataset (2020)* (utoljára megnyitva és használva: 2024.12.03.)
https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset?select=healthcare-dataset-stroke-data.csv

Fejlesztők: *(Budapesti Műszaki és Gazdaságtudományi Egyetem)*

Árvai Péter (főszerkesztő)
Biebel Botond
Tuba Balázs

Segítettek továbbá:

Fekete András *(IT-Universitetet i København)*