DEBRECENI EGYETEM
INFORMATIKAI KAR

Szakdolgozat

Gépi tanuláson alapuló regressziós modell alkalmazása fogyasztói magatartás előrejelzésére.

Témavezető:

Dr. Harangi Balázs

Egyetemi docens

Készítette:

Madár László Adrián

Gazdaságinformatikus

Debrecen

2024

Tartalomjegyzék:

Bevezetés

Az élelmiszerpiacon az egyes termékek fogyasztásának előrejelzése mindig is központi szerepet játszott a termelés, készletkezelés és értékesítés optimalizálása szempontjából. A technológiai fejlődés révén a gépi tanulás (machine learning) eszköztára széleskörű lehetőségeket kínál ezen előrejelzési feladatok megoldására. A gépi tanulás lehetővé teszi olyan összefüggések feltárását a történelmi adatokban, amelyeket hagyományos statisztikai módszerekkel nehéz lenne felismerni, ezáltal pontosabb és megbízhatóbb előrejelzések készíthetők. Szakdolgozatomban célom, hogy egy ilyen gépi tanuláson alapuló megközelítést alkalmazzak a fagylaltfogyasztás előrejelzésére.

A fagylaltfogyasztás egy olyan élelmiszerfogyasztási kategória, amelyet jelentős mértékben befolyásolhatnak az időjárási tényezők, valamint az egyéb gazdasági és környezeti változók. Mivel a fogyasztás jellemzően szezonális jellegű, a nyári hónapokban jelentősen megnő, míg a téli hónapokban csökken, ezért rendkívül érdekes vizsgálni azokat a tényezőket, amelyek meghatározzák a változásokat. Az időjárás, különösen a hőmérséklet és a napsütéses órák száma közvetlen hatással lehet a fagylaltfogyasztásra. Emellett a fogyasztási szokásokat befolyásolhatja a gazdasági környezet, mint például az üzemanyagárak vagy a népességi adatok, amelyek a piaci keresletet befolyásolják.

Szakdolgozatom középpontjában egy olyan regressziós modell fejlesztése áll, amely képes előrejelezni a fagylaltfogyasztás havi változásait különböző időjárási tényezők, üzemanyagárak, CO2-kibocsátás és népességi adatok alapján. Ehhez a feladathoz a *Python* programozási nyelvet használtam. A modell felépítéséhez olyan elismert *Python* könyvtárakat alkalmazok, mint a *Pandas, NumPy, Statsmodels, Tensorflow* és a *Scikit-learn.* Ezek a könyvtárak lehetővé teszik az adatok hatékony feldolgozását és a különböző gépi tanulási algoritmusok implementálását, amelyekkel megbízható predikciókat állíthatok fel.

Az adatok elemzésének folyamán különösen fontos a vizualizációk készítése, amelyek segítségével a különböző tényezők hatásait látványosan és érthetően tudom bemutatni. A *matplotlib* és *seaborn* könyvtárak használata ebben nyújt majd támogatást, hiszen ezek segítségével grafikusan ábrázolhatóak a különböző változók közötti kapcsolatok és trendek. Ezek az ábrák nem csupán a szakdolgozat olvasóinak nyújtanak könnyebben értelmezhető képet a kutatási eredményekről, hanem segítenek a modellek finomhangolásában és értékelésében is.

A téma relevanciája

A fogyasztás előrejelzése kiemelkedő fontossággal bír a kereskedelmi és fogyasztói szektorban. Egy jól működő előrejelzési modell segítségével a vállalatok pontosabban tervezhetik meg termelési kapacitásaikat, optimalizálhatják raktárkészleteiket, és gyorsabban tudnak reagálni a piaci változásokra. A fogyasztási szokások előrejelzése különösen fontos olyan szezonális termékek esetében, mint a fagylalt, hiszen ezeknél a termékeknél a kereslet jelentős ingadozást mutat az év során. Egy olyan modell kifejlesztése, amely képes a szezonális szintű változásokat pontosan előrejelezni, nemcsak gazdasági-, hanem fenntarthatósági szempontból is jelentős.

A gépi tanulás alkalmazása a fogyasztás előrejelzésére a modern technológia fejlődésének köszönhetően egyre elterjedtebbé válik. Az ilyen modellek segítségével a vállalatok nemcsak a fogyasztói igények pontosabb kielégítésére, hanem az erőforrások hatékonyabb feldolgozására is képesek.

Emellett a gazdasági változók, mint az üzemanyagárak és a népesség, szintén fontos szerepet játszanak a piaci dinamika megértésében. Az üzemanyagárak például közvetett módon befolyásolhatják a fagylalt termelési és szállítási költségeit, ami a végső fogyasztói árakra és ezáltal a keresletre is hatással lehet. A népesség alakulása pedig közvetlenül hat a piaci keresletre, hiszen a nagyobb népesség nagyobb piacot is jelent.

Az elvégzendő munkáról:

A szakdolgozatom során első lépésként egy adatbázis összeállítása szükséges, amely tartalmazza a jégkrémfogyasztás havi adatait, valamint az időjárási és gazdasági tényezőket. Az adatokat különböző forrásokból gyűjtöm. Az adatok előkészítését és tisztítását követően vizualizációs eszközökkel feltárom a lehetséges kapcsolatokat az attribútumok között majd megkezdem a modellek fejlesztését, amelyek segítségével különböző regressziós és idősoros módszereket alkalmazok a fogyasztás előrejelzésére. Az előrejelzési pontosság érdekében a modelleket finomhangolom, és az eredményeket különböző teljesítménymérő eszközökkel értékelem, mint például a determinációs együttható (R^2), az átlagos négyzetes hiba négyzetgyöke (RMSE), az átlagos abszolút hiba (MAE), valamint az átlagos hibával, ami az RMSE és a MAE átlaga.

A dolgozat második felében összehasonlítom a lineáris, nem lineáris és idősoros modellek eredményeit, és értékelem azok előrejelzési teljesítményét. A vizsgálat során figyelmet fordítok arra, hogy a különböző változók hogyan befolyásolják a fagylaltfogyasztás előrejelzését, és milyen szerepet játszanak az időjárási, gazdasági és környezeti tényezők ebben a folyamatban.

1. Az adatok forrása és bemutatása:

1.1. Az adatforrások bemutatása:

Az adatkészleteket több forrásból gyűjtöttem össze, hogy a különböző változók széles spektrumát lefedjem. A földgázárak és a CO_2-kibocsátási adatok az *U.S. Energy Information Administration (EIA)* adatbázisából származnak, míg a népességi adatokat a *Worldometers* szolgáltatta. A fagylaltfogyasztási adatokat a *Federal Reserve Bank of St. Louis (FRED)* gyűjtötte össze. Az egyetlen kivételt az időjárási adatok jelentik, amelyekhez nem áll rendelkezésre hivatalos licenc, így hitelességük nem teljesen biztosított.

A használt adatbázisok mindegyike az Amerikai Egyesült Államokból származik, azonban időbeli terjedelmük eltérő. A CO_2-kibocsátási adatokat havi bontásban rögzítették, míg a népességi adatok éves szinten. A fagylaltfogyasztás szintén havi gyakorisággal áll rendelkezésre, az időjárási és a földgázárak adatai pedig napi szintűek. Bár ezek az időbeli eltérések kihívást jelentenek az elemzés során, léteznek közös időszakok, amelyek lehetővé teszik a különböző adatforrások összehangolását a modellezéshez.

1.2. Az adatok bemutatása:

Az elemzéshez használt adattáblák a következők: Időjárás – weather, Földgázárak – gas, Szén-dioxid kibocsájtás – co2, Népességi adatok – population és végül a fagylaltfogyasztás – ice_cream. Az adattábláink attribútumait részletesebben bemutatja az alábbi táblázat:

Weather (7 város külön CSV-ben) [1]		
	date_time	Dátumok napi gyakorisággal
	maxtempC	Maximum hőmérséklet
	mintempC	Minimum hőmérséklet
	totalSnow_cm	Összes hó centiméterben
	sunHour	Napos órák száma
	uvIndex	UV index
	uvIndex.1	UV index (valamilyen másfajta mérés, de nem tudjuk milyen)
	moon_illumination	A hold láthatósága
	moonrise	Holdfelkelte (string)
	moonset	Holdnyugta (string)
	sunrise	Napfelkelte (string)
	sunset	Napnyugta (string)
	DewPointC	Harmatpont
	FeelsLikeC	Hőérzet
	HeatIndexC	Hőindex
	WindChillC	Szélhűtés – szél hatására érződő hőmérséklet
	WindGustKmph	Széllökés Km/h
	cloudcover	Felhő lefedettség, felhőzet
	humidity	Páratartalom
	precipMM	Csapadék mm-erben.
	pressure	Légnyomás
	tempC	Átlag hőmérséklet
	visibility	Láthatóság
	winddirDegree	Szélirányfok: 0° - Északi szél
	windspeedKmph	Szélsebesség Km/h
Földgáz – gas [2]		
	Date	Dátum napi gyakorisággal.
	Price	Aznapi átlag földgáz ár.

Széndioxid kibocsájtás – co2 [3]	
Month	Dátum havi gyakorisággal.
Total Energy CO2 Emissions	Millió metrikus tonna szén-dioxid
Népesség – population [4]	
Year	Év
Population	Népesség
Yearly % Change	Százalékos népességváltozás az előző időszakhoz képest.
Yearly Change	Népesség növekedés vagy csökkenés abszolút értéke.
Migrants (net)	Nettó migráció
Median Age	Medián életkor
Fertility Rate	Termékenységi ráta
Density (P/Km²)	Népsűrűség
Urban Pop %	Városi népesség százalélos aránya
Urban Population	Városi népesség abszolút mennyisége
Country's Share of World Pop	Az ország népessége százalékosan kifejezve a világ népességéhez képest.
World Population	A világ népessége
U.S. Global Rank	Az egyesült államok népesség szerinti rangja a világban.
Fagylaltfogyasztás – ice_cream [5]	
DATE	Dátum havi gyakorisággal.
IPN31152N (value)	A hónapban mért fagylaltfogyasztás millió gallonban mérve.

1. Táblázat – Az adattáblák bemutatása attribútumaik értelmezése – saját szerkesztés.

1.3. Az adatok integrálása:

Az adatok összegyűjtése és integrálása során az egyes adatforrások elsődleges kulcsa a dátum attribútum volt, így az adattáblák összeillesztése a dátumok mentén történt. Mivel az adatforrások időbeli terjedelme és a gyűjtés gyakorisága eltérő volt, először meg kellett vizsgálnom,

mely időszakokban hangolhatók össze az adatok. Az alábbi táblázat összefoglalja az egyes adatforrások időtartamait és gyakoriságát:

Adathalmaz:	Időtartam:	Gyűjtés gyakorisága:
population	1955 - 2020	Éves adatok
ice_cream	**1972-01-01** - 2024-03-01	Havi adatok
co2	1973.01 - 2024-06-01	Havi adatok
weather	**2008-07-01 - 2019-12-31**	Napi adatok
gas	1997-01-07 - 2020-09-01	Napi adatok

2. Táblázat – Az adattáblák tartományelemzése – saját szerkesztés.

Az illesztési lehetőségek közül két fő megközelítést választottam:

1. **Weather adattáblához való illesztés:** Itt a független változókat a városok napi időjárási adatainak mentén integráltam a 2008-07-01 - 2019-12-31 közötti időszakban, mivel minden adatforrás rendelkezett adatokkal ezen időszakban. Az előforduló hiányzó értékeket súlyozott átlagolással töltöttem fel a konzisztensebb eredmények érdekében.
2. **Ice_cream adattáblához való illesztés:** Ennél a módszernél az országos gazdasági és környezeti változókat a fagylaltfogyasztási adatokhoz illesztettem, az időjárási adatok városi átlagát hozzárendelve a fogyasztási napokhoz. Az ice_cream adatbázis időtartománya nem minden adatforrás számára volt megfelelő, ezért az integrálási időszakot 2008-07-01 - 2019-12-01 közé szűkítettem.

Elsődleges kulcs egységesítése: Az adatforrásokban található különböző dátumformátumokat és időbeli gyakoriságokat egységesítettük az **YEAR**, **MONTH**, és **DAY** formátumra, így biztosítva a konzisztens illesztést. A különböző dátumformátumok egységesítésével lehetővé vált a modellezéshez szükséges adatok összekapcsolása és az elemzések előkészítése.

1.4. Adattáblák összeillesztése:

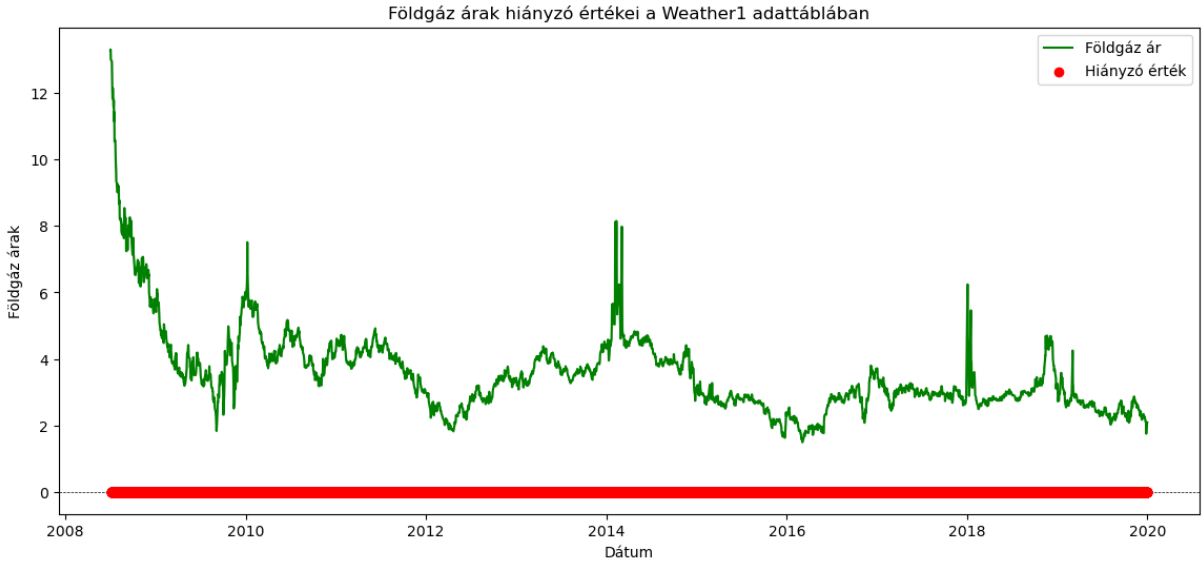
A célom kettő adathalmaz létrehozása melyek a *weather1* és az *ice_cream1*, melyek a végső kimentendő adattáblák lesznek.

Először a *weather* adattáblát kellett létrehoznom, mivel az adatok városonként külön CSV fájlokban voltak tárolva. Mivel mindegyik CSV fájl azonos oszlopstruktúrával rendelkezett, a városok adatai könnyen integrálhatók voltak. Az egyesítés során a *weather* adattábla YEAR, MONTH, DAY és CITY attribútumokat kapott, melyek elsődleges kulcsként szolgálnak. A teljes adatkészlet kilenc várost tartalmaz: Chicago, Houston, Los Angeles, Miami, Montreal, New York, San Francisco, Toronto (Kanada) és Vancouver (Kanada). Az integrálás során csak az Amerikai Egyesült Államok városainak adatait használtam fel. Az alábbi kód végzi el a *weather* adattábla előállítását:

```
# Először beolvassuk a chicago.csv fájlt:
weather = pd.read_csv('Adatok/weather/chicago.csv')

# A 'date_time' oszlopot dátumidő formátummá alakítjuk
weather['date_time']  =  pd.to_datetime(weather['date_time'],
format='%Y-%m-%d')

# Új oszlopok létrehozása a chicago adatokhoz
weather['YEAR'] = weather['date_time'].dt.year
weather['MONTH'] = weather['date_time'].dt.month
weather['DAY'] = weather['date_time'].dt.day
weather['CITY'] = 'Chicago'

# A többi város, amelyet hozzá fogunk adni a chicago DataFrame-
hez
cities = ['houston', 'la', 'miami', 'montreal', 'ny', 'sf']

# A többi város adatainak beolvasása és feldolgozása
for city in cities:
    # Beolvassuk a város adatfájlját
    df_city = pd.read_csv(f'Adatok/weather/{city}.csv')

    # A 'date_time' oszlopot dátumidő formátummá alakítjuk
```

```
    df_city['date_time']  =  pd.to_datetime(df_city['date_ti-
me'], format='%Y-%m-%d')

    # Új oszlopok létrehozása a város adataihoz
    df_city['YEAR'] = df_city['date_time'].dt.year
    df_city['MONTH'] = df_city['date_time'].dt.month
    df_city['DAY'] = df_city['date_time'].dt.day
    df_city['CITY'] = city.capitalize()  # A város nevét nagy
kezdőbetűvel adjuk hozzá

    # Az aktuális város adatainak hozzáadása a chicago
DataFrame-hez
    weather = pd.concat([weather, df_city], ignore_index=True)

# A 'date_time' oszlop átnevezése 'DATE' névre
weather.rename(columns={'date_time': 'DATE'}, inplace=True)

# Az oszlopok sorrendjének módosítása
weather = weather[['DATE', 'YEAR', 'MONTH', 'DAY', 'CITY'] +
[col for col in weather.columns if col not in ['DATE', 'YEAR',
'MONTH', 'DAY', 'CITY']]]

# Az első 10 sor megjelenítése szép táblázatos formában
display(weather.head(10))
```

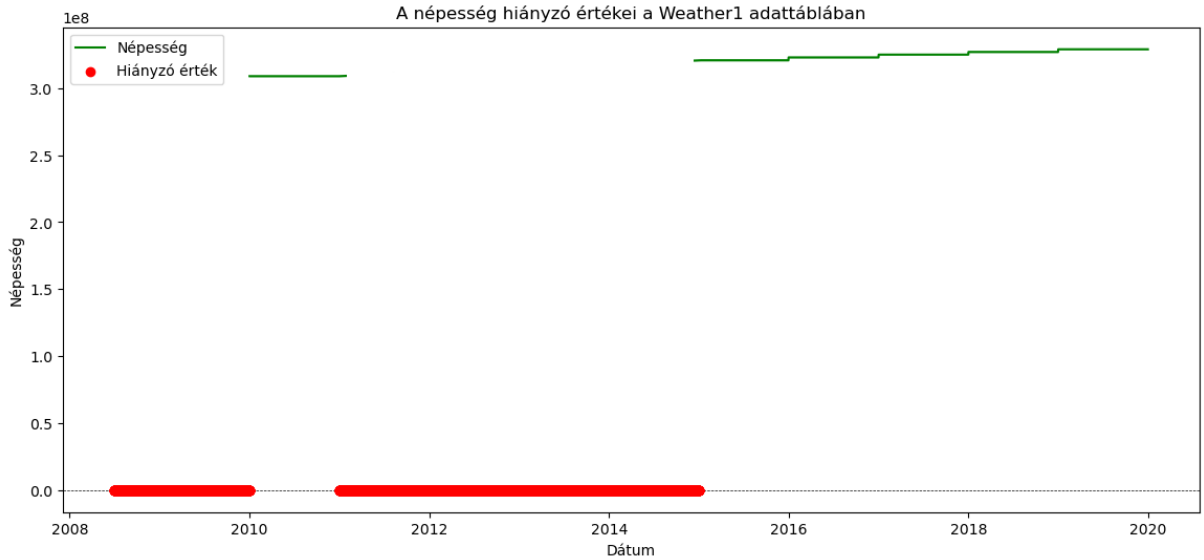
A *weather1* végső adattábla második lépéseként a földgázárakat (*gas*) illesztettem a *weather* adattáblához. Mindkét adatforrás napi bontású, így az összeillesztés során nem volt szükség bonyolultabb átalakításokra. Az illesztés azonban pontosan 9009 hiányzó értéket eredményezett, mivel a földgázárak nem minden napra álltak rendelkezésre. Az 1. ábra bemutatja a hiányzó értékek eloszlását, ahol jól látható, hogy a hiányzó adatok azokhoz az időpontokhoz kapcsolódnak, amelyekhez a földgáz napi áradatai nem lettek feljegyezve.



1. ábra - A földgázárak hiányzó értékei - saját szerkesztés.

Ezután a CO_2-kibocsátási (*co2*) adatokat illesztettem a *weather1* adattáblához. Mivel havi adatokat kellett napi adatokhoz igazítani, ezt úgy oldottam meg, hogy minden naphoz a megfelelő hónap azonos értékét rendeltem hozzá. Ezzel a módszerrel az illesztés során nem keletkeztek hiányzó értékek.

A következő lépésben a népességi (*population*) adatokat illesztettem a *weather1* adattáblához, ahol éves adatok alapján dolgoztam: minden évhez a megfelelő népességi értéket rendeltem hozzá. Mivel nem minden évhez állt rendelkezésre adat, az illesztés 14 070 hiányzó értéket eredményezett. A 2. ábra szemlélteti a hiányzó értékek eloszlását. A *population* adattáblából kizárólag a *population* oszlopot használtam fel.



2. ábra – A népesség hiányzó értékei – saját szerkesztés.

Végül a fagylaltfogyasztási (*ice_cream*) adatokat illesztettem a *weather1* adattáblához. Itt havi adatokat rendeltem hozzá a napi adatokhoz, hasonlóan a CO_2-kibocsátás illesztéséhez, azaz minden naphoz az adott hónap értékeit rendeltem. Ez az illesztés nem eredményezett hiányzó értéket.

Az *ice_cream1* végső adattábla összeállítása összetettebb feladat volt. Kiindulásként a *weather* adattáblát használtam, amelyet havi szintű adatokra aggregáltam. Először városonként szűrtem az adatokat, majd egyenként aggregáltam őket havi gyakoriságra. A *weather* adattábla számos különböző attribútumot tartalmazott, amelyeket eltérő módszerekkel vontam össze: egyes változókat átlagolással, másokat minimum-, maximumértékekkel, illetve összegzéssel aggregáltam. Így a végső adattábla a hét város havi szinten összesített adatait tartalmazta. Az alábbi kód mutatja be az aggregálást:

```
# Az összes város lekérése a weather dataframe-ból
cities = weather["CITY"].unique()

# Dictionary, amelyben a városokhoz rendelt ice_cream dataframe-
ek lesznek
ice_cream_cities = {}

# Az alsó és felső dátumhatárok
lower_cutoff_date = pd.Timestamp(year=2008, month=7, day=1)
upper_cutoff_date = pd.Timestamp(year=2019, month=12, day=1)

# Aggregálási szabályok definiálása
agg_rules = {
    'maxtempC': 'max',
    'mintempC': 'min',
    'totalSnow_cm': 'sum',
    'sunHour': 'sum',
    'uvIndex': 'mean',
    'moon_illumination': 'mean',
    'DewPointC': 'mean',
```

```
    'FeelsLikeC': 'mean',
    'HeatIndexC': 'mean',
    'WindChillC': 'mean',
    'WindGustKmph': 'mean',
    'cloudcover': 'mean',
    'humidity': 'mean',
    'precipMM': 'sum',
    'pressure': 'mean',
    'tempC': 'mean',
    'visibility': 'mean',
    'winddirDegree': 'mean',
    'windspeedKmph': 'mean',
    # moonrise, moonset, sunrise, sunset nem lesznek az aggregá-
lás részei
}

# Minden városhoz: napi időjárási adatok havi átlagolása és ösz-
szefésülés az ice_cream adatokkal
for city in cities:
    # Az adott városhoz tartozó ice_cream dataframe-et hozzáren-
delése
    ice_cream_cities[city] = ice_cream.copy()

    # Szűrjük a weather dataframe-et az adott városra
    city_weather = weather[weather['CITY'] == city]

    # Kiválasztjuk a numerikus oszlopokat
    numeric_cols = city_weather.select_dtypes(inc-
lude=['number']).columns
    # Csoportosítás YEAR, MONTH alapján
    weather_monthly_agg = city_weather.groupby(['YEAR',
'MONTH']).agg(agg_rules).reset_index()
```

```
    # Merge-elés az ice_cream városhoz tartozó adatainak YEAR, MONTH alapján
    ice_cream_cities[city] = pd.merge(ice_cream_cities[city], weather_monthly_agg, on=['YEAR', 'MONTH'], how='left')

    # Szűrés a dátumok alapján (alsó és felső dátumhatár)
    ice_cream_cities[city] = ice_cream_cities[city][(ice_cream_cities[city]['DATE'] >= lower_cutoff_date) & (ice_cream_cities[city]['DATE'] <= upper_cutoff_date)]

    # Elhagyjuk a nem szükséges oszlopokat
    ice_cream_cities[city] = ice_cream_cities[city].drop(columns=['DAY_x', 'DAY_y'], errors='ignore')

# Például egy város (pl. 'New York') adatai
display(ice_cream_cities['New York'].head(10))
```

```
# Üres lista a DataFrame-ek tárolásához
dataframes = []
# Végigmegyünk a dictionary-n
for city, df in ice_cream_cities.items():
    # Hozzáadjuk a CITY oszlopot az adott DataFrame-hez
    df['CITY'] = city
    # Hozzáadjuk a DataFrame-t a listához
    dataframes.append(df)
# Összeillesztjük az összes DataFrame-t egyetlen DataFrame-be
ice_cream1 = pd.concat(dataframes, ignore_index=True)

# Ellenőrizzük az eredményt
display(ice_cream1.head())
```

Az *ice_cream1* adattáblához ezután hozzáillesztettem a földgázárakat (*gas*). Mivel napi adatokat csatoltam havi adatokhoz, előbb átlagoltam a napi értékeket, hogy összhangba hozzam a havi szinttel. Ez az illesztés végül nem eredményezett hiányzó értékeket.

Ezután a CO2 kibocsátás (*co2*) adatokat illesztettem az *ice_cream1* adattáblához. Mivel mindkét adattábla havi szintű adatokat tartalmazott, nem volt szükség további transzformációra. Az illesztés során nem keletkeztek hiányzó értékek.

Végül a népességi (*population*) adatokat illesztettük az *ice_cream1* adattáblához. Az illesztés 462 hiányzó értéket generált. Legvégül az ice_cream1 adattáblához hozzá illesztettük az *ice_cream* adattáblát is.

2. Az adatok tisztítás és előfeldolgozása:

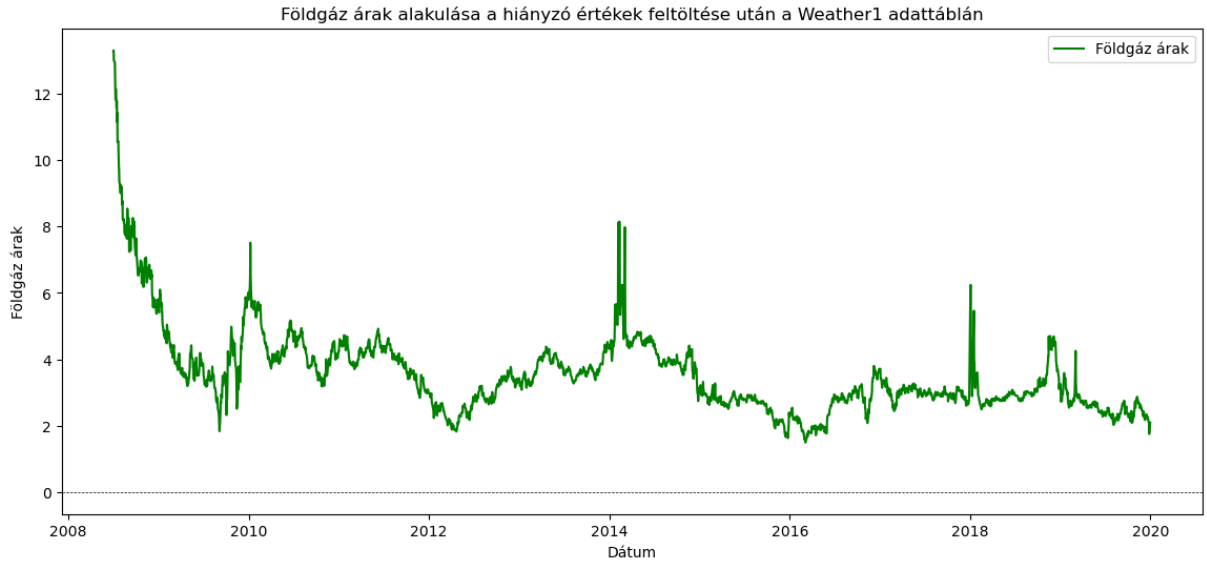
2.1. A hiányzó adatok kezelése:

A két elkészült adattáblában, a *weather1* és az *ice_cream1* adattáblákban is előfordultak hiányzó értékek. A *weather1* adattáblában a földgázárak és a népességi adatok tartalmaztak hiányzó értékeket. A földgázárakat úgy kezeltem, hogy a két legközelebbi meglévő értéket figyelembe véve átlagoltam őket, ügyelve arra, hogy ha egymás után több hiányzó érték is szerepelt, akkor azokat egyenletesen pótoljam. Az alábbi kóddal töltöttem fel a földgázárak hiányzó értékeit:

```
# Iteráljuk a DataFrame sorait
for index in range(len(weather1)):
    if pd.isnull(weather1.loc[index, 'Gas price']):
        # Keresd meg az előző nem hiányzó értéket
        prev_index = None
        for i in range(index - 1, -1, -1):
            if pd.notnull(weather1.loc[i, 'Gas price']):
                prev_index = i
                break
        # Keresd meg a következő nem hiányzó értéket
        next_index = None
        for i in range(index + 1, len(weather1)):
            if pd.notnull(weather1.loc[i, 'Gas price']):
                next_index = i
                break
        # Ha megtaláltuk mindkettőt, számítsuk ki az átlagot és
töltsük ki a hiányzó értéket
        if prev_index is not None and next_index is not None:
            prev_value = weather1.loc[prev_index, 'Gas price']
            next_value = weather1.loc[next_index, 'Gas price']
            # A két érték közötti teljes különbség
            total_diff = next_value - prev_value
            # A köztes értékek száma
            num_missing = next_index - prev_index - 1
            # Lépésköz kiszámítása, hogy egyenletesen eloszthassuk
a különbséget
```

```
            step = total_diff / (num_missing + 1)
            # Töltsük ki a hiányzó értékeket egyenletes lépésközök-
kel
            for i in range(prev_index + 1, next_index):
                weather1.loc[i, 'Gas price'] = prev_value + (i -
prev_index) * step

# Az eredmény kiírása
display(weather1)
```

A 3. ábra szemlélteti a feltöltött adatkészletet:



3. ábra – A földgázárak a hiányzó értékek feltöltése után – saját szerkesztés.

A *weather1* adattáblában a népességi adatokból a 2008-as év alsó határának adatát nem találtam, ezért azt külső forrásból szereztem be. [15]

Ezt követően, hasonlóan a földgázárakhoz, átlagolással pótoltam az adatokat, figyelve a kiegyensúlyozott feltöltésre. Az *ice_cream1* adattáblában csupán a népességi adatoknál fordult elő hiányzó érték, amelyet egyszerűen átemeltem a *weather1* adattáblából.

2.2. A kiugró értékek észlelése és kezelése:

A kiugró értékeket az interkvartilis távolság (IQR) másfélszeresével határoztam meg, és mindkét végső adattáblán az alábbi szabályokat alkalmaztam:

- Az időjárási mutatók esetében csak azokat a kiugró értékeket távolítom el, amelyek az interkvartilis távolság kétszeresén kívül esnek. Kivételt képez a *'totalSnow_cm'* attribútum, mivel ennek szigorú szűrése az adatok nagy részének törlését eredményezné ebből az oszlopból.
- A *Gas price* (üzemanyagár) kiugró értékeit az interkvartilis távolság másfélszerese alapján szűröm.
- A *co2 emission* (CO_2 kibocsátás) esetén a kiugró értékeket nem kezelem.

Ez a módszer lehetővé teszi, hogy kizárólag az extrém időjárási adatokat távolítsam el, és az üzemanyagár esetén a jelentős áringadozásokat kezeljem, amelyek hosszú távon torzítanák a modelljeimet. A CO_2 kibocsátás változásait megtartjuk, mivel ezek hosszú távon fontos hatással lehetnek a környezetre.

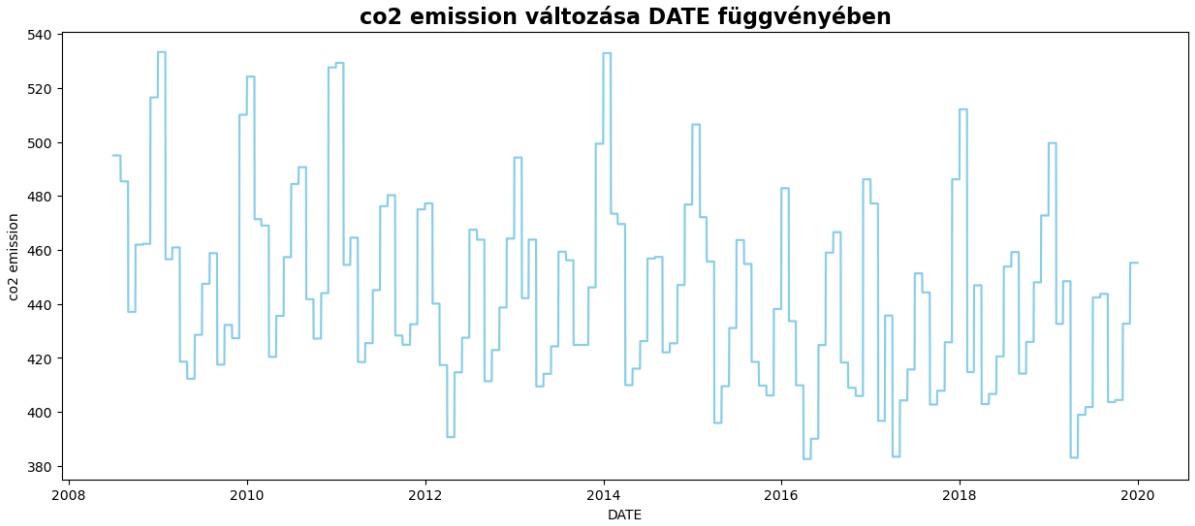
Mindkét adattábla eredeti formáját megőriztem, és létrehoztam a kiugró értékektől mentes változatokat az alábbi fájlneveken: *weather1_kiugro_ertekek_nelkul.csv* és *ice_cream1_kiugro_ertekek_nelkul.csv*. A modellek felépítésekor, finomhangolásakor, ha szükséges akkor ezen adattáblákkal végzem el a betanítást.

3. Adatok vizualizálása:

Szakdolgozatomban különböző vizualizációs eszközöket alkalmaztam az adatok közötti kapcsolatok szemléltetésére. Az egyszerűbb típusú ábrák, mint az *oszlop-* és *vonaldiagramok*, valamint a *dobozábrák* és *pontdiagrammok* hasznosnak bizonyultak az adatok alapvető eloszlásainak és összefüggéseinek bemutatásában. A komplexebb vizualizációk, például a *hexbin* és a *KDE (Kernel Density Estimate)* ábrák segítettek a sűrűségi eloszlások pontosabb ábrázolásában, több változó közötti eloszlások vizualizálására a *páros ábra (pairplotot)* és a *hőtérkép (heatmap)* használtam, amely lehetővé tette az attribútumok közötti komplex kapcsolatok könnyebb megértését. Minden vizualizációt mindkét adathalmazon, a *weather1* és az *ice_cream1* adattáblákon is lefuttattam, hogy az adatok közötti kapcsolatokat mindkét esetben átfogóan és részletesen feltérképezhessem.

3.1. Időjárási és gazdasági változók vizualizációja:

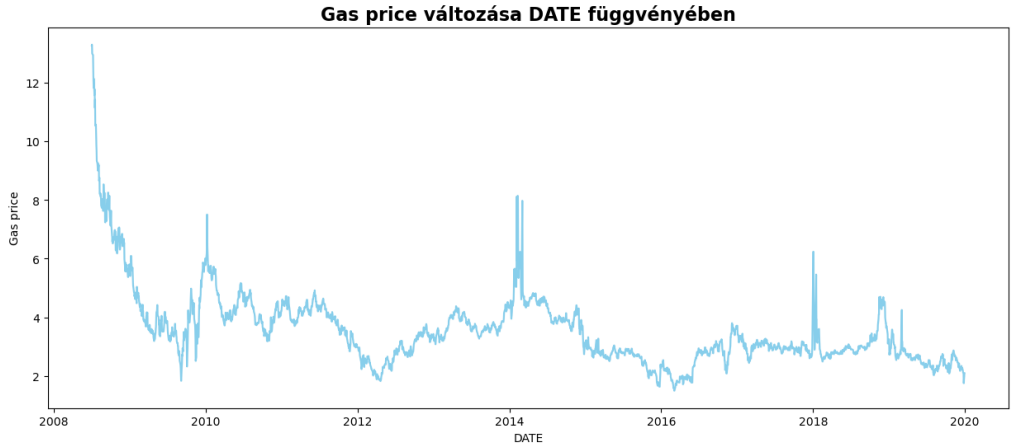
Széndioxid kibocsájtás: (4. ábra) Mivel az eredeti adatforrásunkban havi adataink voltak, ezért a két adatforrás vizualizációja között nincs különbség.



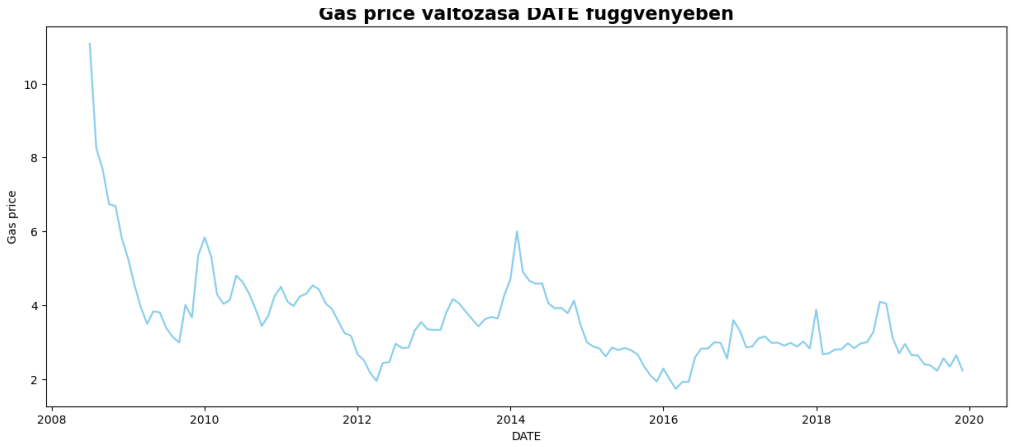
4. ábra – A széndioxid kibocsájtás mértéke az évek során – saját szerkesztés.

Az ábra alapján szezonalitást és egy enyhe csökkenő tendenciát feltételezhetünk.

A földgáz árak az alábbiak szerint alakultak az évek során: Napi árak (5. ábra), majd havonta átlagolt árak (6. ábra)

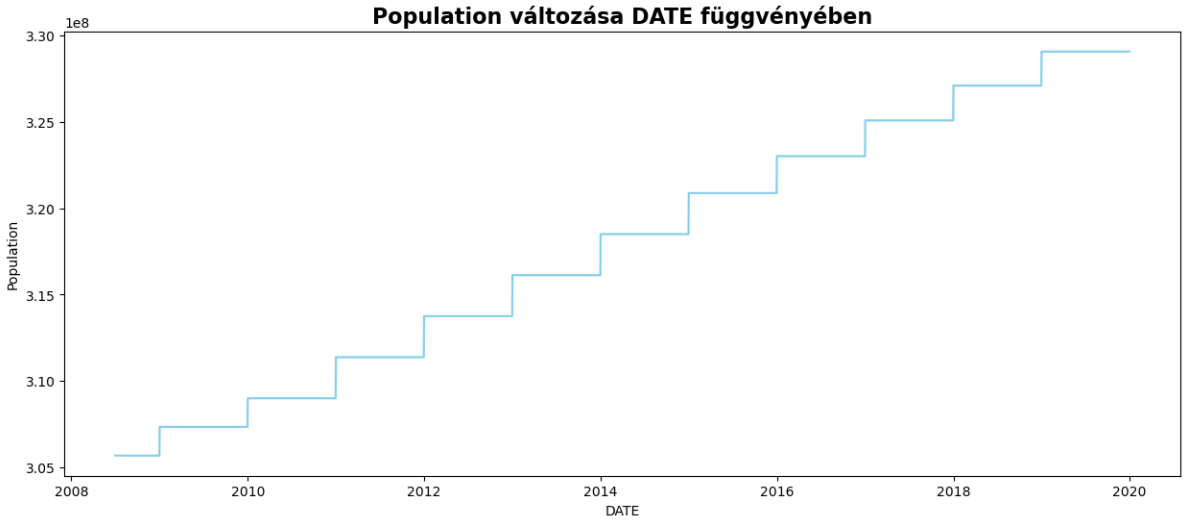


5. ábra – Földgáz árak alakulása az évek során (napi bontás) – saját szerkesztés.



6. ábra – Földgáz árak alakulása az évek során (havi bontás) – saját szerkesztés.

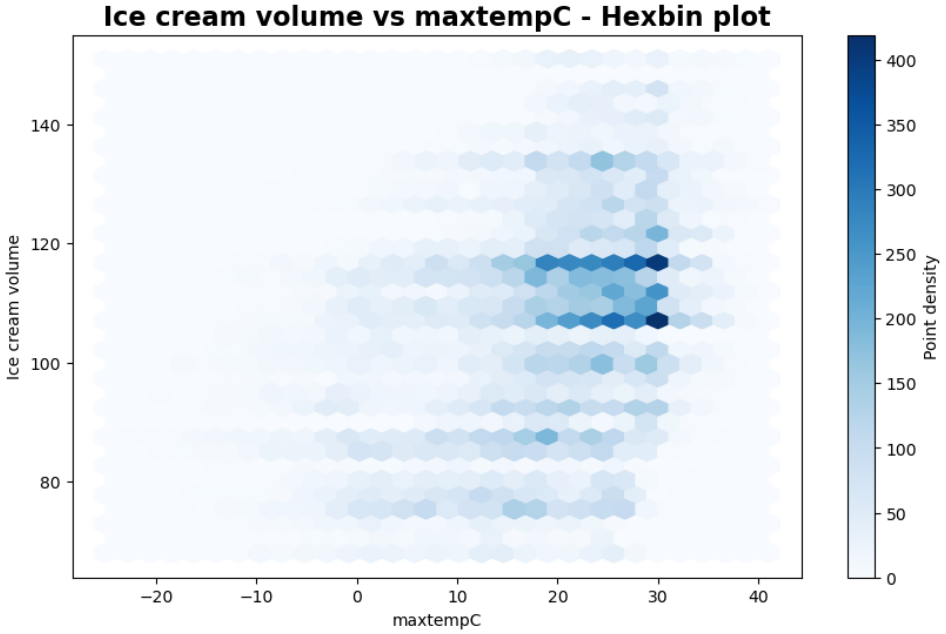
A népességi adatok (7. ábra) mind a kettő adattáblában a következőképpen alakultak az évek során:



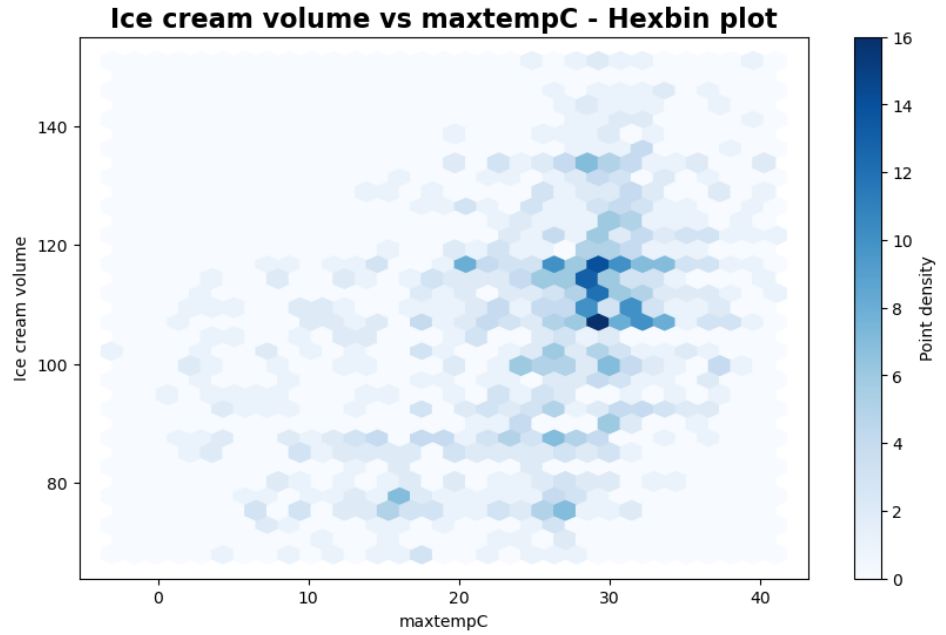
7. ábra – A népesség változása az évek folyamán – saját szerkesztés.

3.2. Két- és többváltozós ábrázolások:

A maximum hőmérséklet és a fagylaltfogyasztás kapcsolata – *weather1* (8. ábra) majd *ice_cream1* (9. ábra) adathalmazokon:



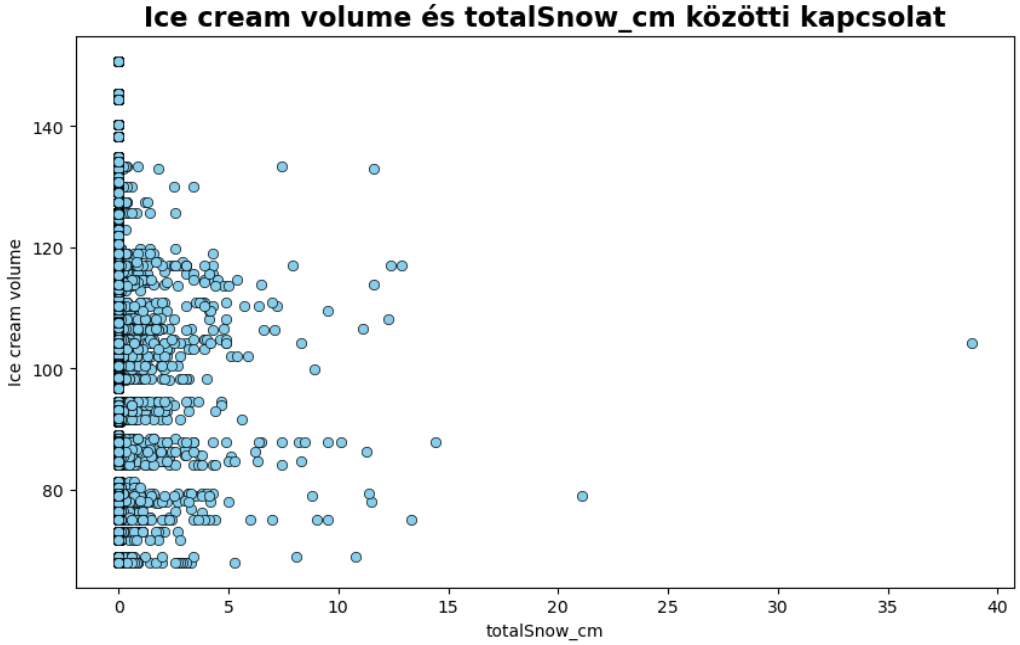
8. ábra – A maximum hőmérséklet és a fagylaltfogyasztás együttes eloszlása (napi bontás) – saját szerkesztés.



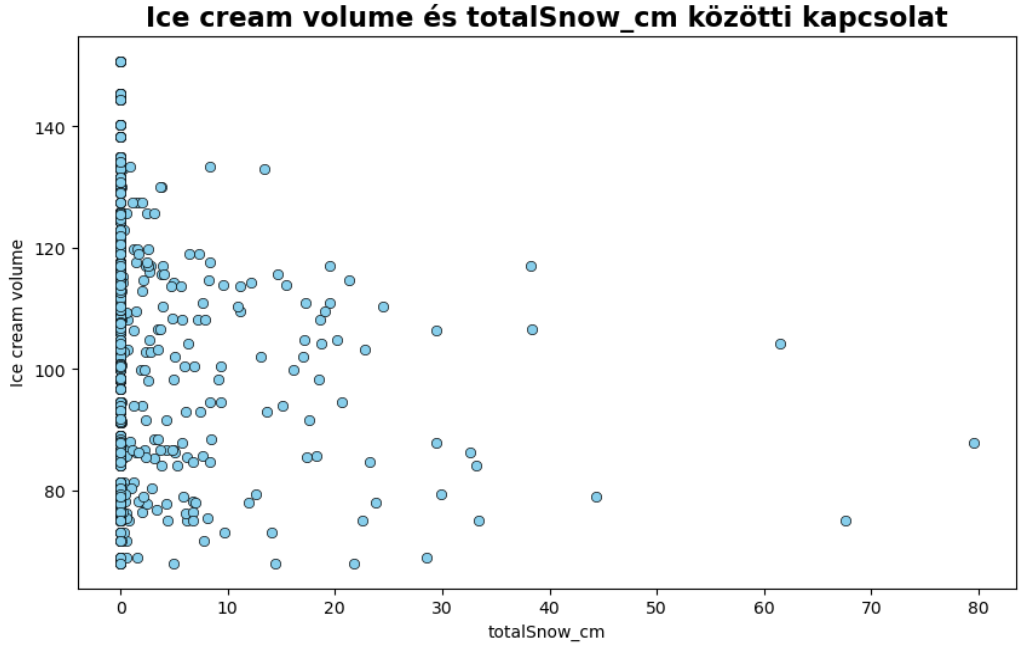
9. ábra – A maximum hőmérséklet és a fagylaltfogyasztás együttes eloszlása (havi bontás) – saját szerkesztés.

Az ábrák alapján jól megfigyelhető a két változó közötti összefüggés: a fogyasztás sűrűsödése körülbelül 30 °C körüli hőmérsékleten, 115 millió gallon környékén figyelhető meg. Alacsony maximális hőmérséklet esetén a fogyasztás is alacsonyabb, míg rendkívül magas hőmérséklet esetén emelkedést tapasztalunk a fogyasztásban.

Az összes lehullott hó és a fagylaltfogyasztás kapcsolata – *weather1* (10. ábra) majd *ice_cream1* (11. ábra) adathalmazokon:



10. ábra – Az összes lehullott hó és a fagylalfogyasztás eloszlása (napi bontás) – saját szerkesztés.



11. ábra – Az összes lehullott hó és a fagylalfogyasztás eloszlása (havi bontás) – saját szerkesztés.

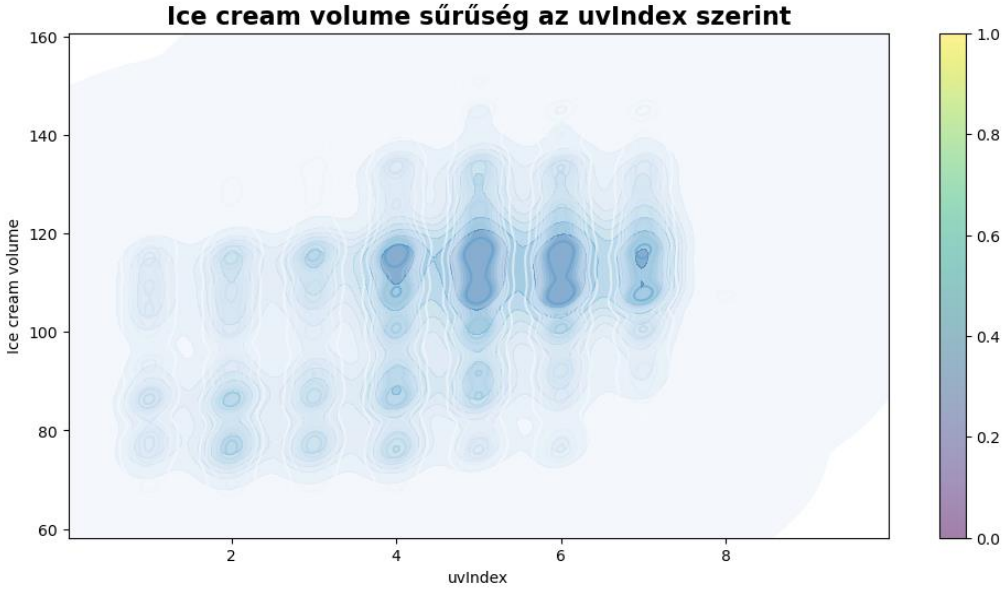
Az ábrán jól látható, hogy a fagylaltfogyasztás betömörül a 0 cm lehullott hó köré, ha van is hó, akkor az ábrákon látható pontok, amik megjelennek azok egy-egy pontok.

A(z) **totalSnow_cm** és **Ice cream volume** korrelációja a weather adathalmazban: **-0.0581**

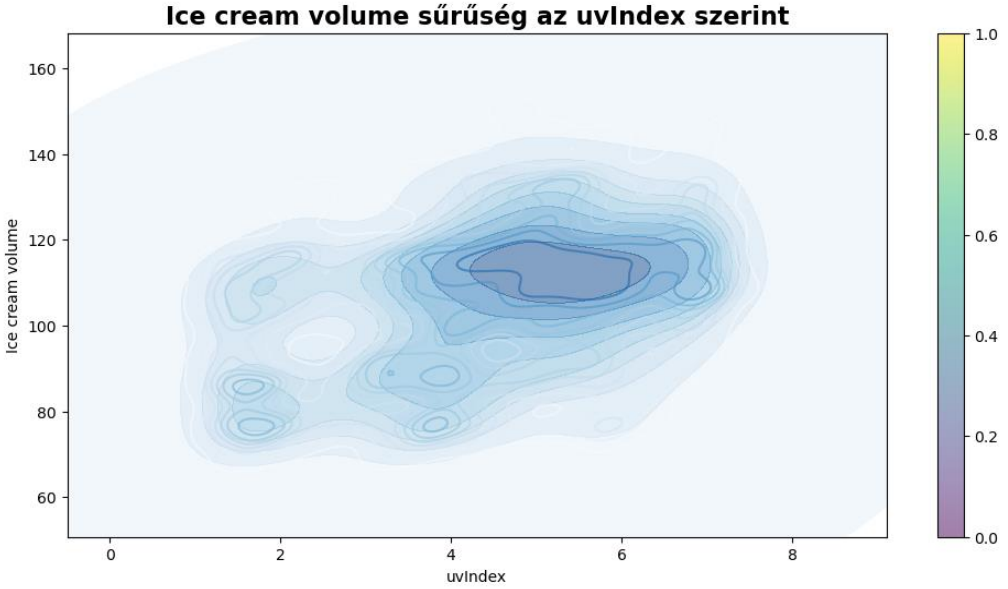
A(z) **totalSnow_cm** és **Ice cream volume** korrelációja az ice_cream adathalmazban: **-0.1650**

Az adataink aggregálásával a **totalShnow_cm** változónak az **Ice cream volume** változóval vett korrelációjának értékét csaknem megháromszoroztuk.

Az uvindex és a fagylaltfogyasztás kapcsolata – *weather1* (12. ábra) majd *ice_cream1* (13. ábra) adathalmazokon:



12. ábra – Az UV index és a fagylaltfogyasztás együttes eloszlása (napi bontás) – saját szerkesztés.



13. ábra – Az UV index és a fagylaltfogyasztás együttes eloszlása (havi bontás) – saját szerkesztés.

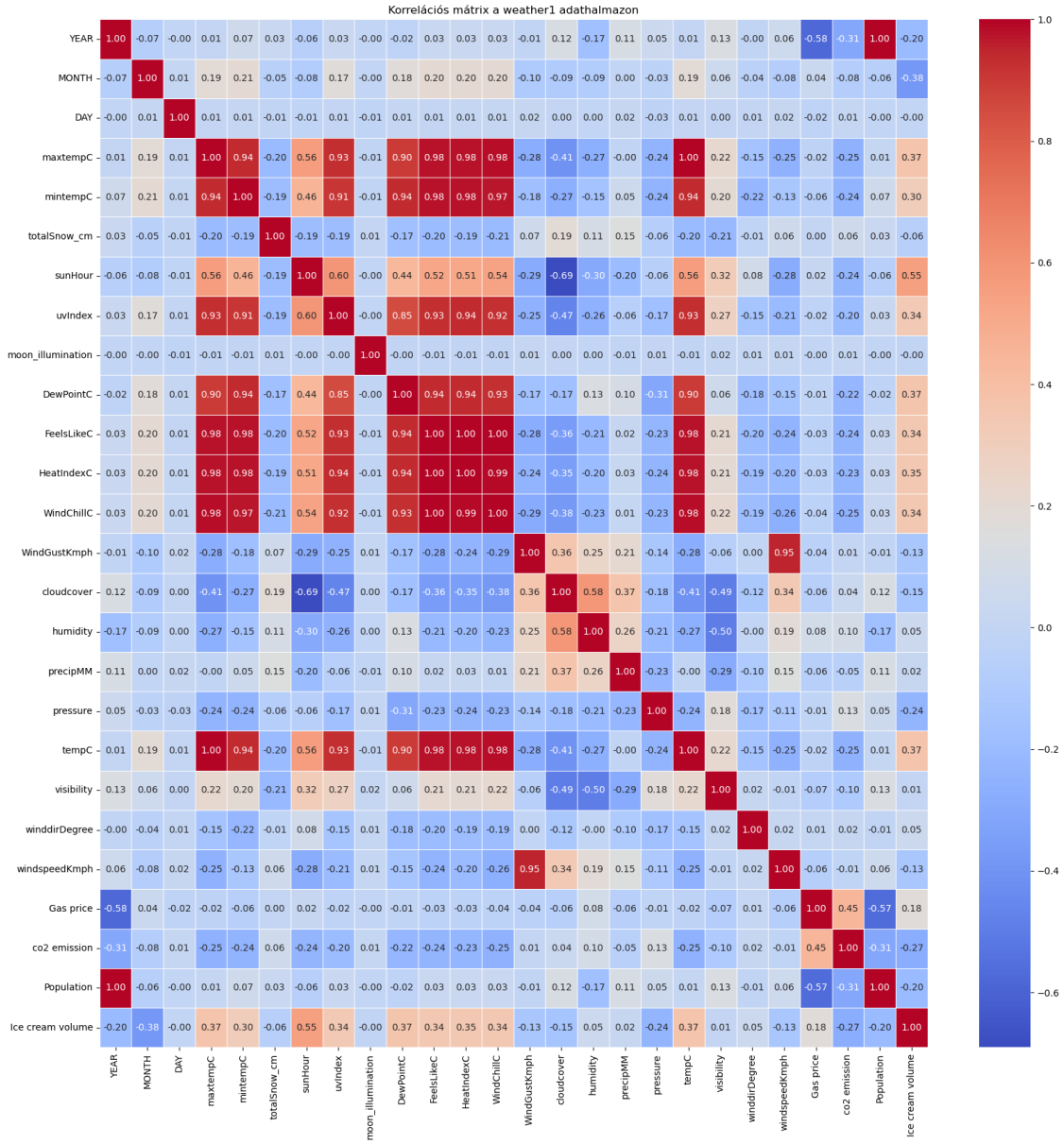
Mind a két esetben az látható, hogy az adatok az UV index 5 értékénél sűrűsödnek, a fogyasztás 115 millió gallonnál.

A(z) **uvIndex és Ice cream volume** közötti korreláció a weather adathalmazban: **0.3364**

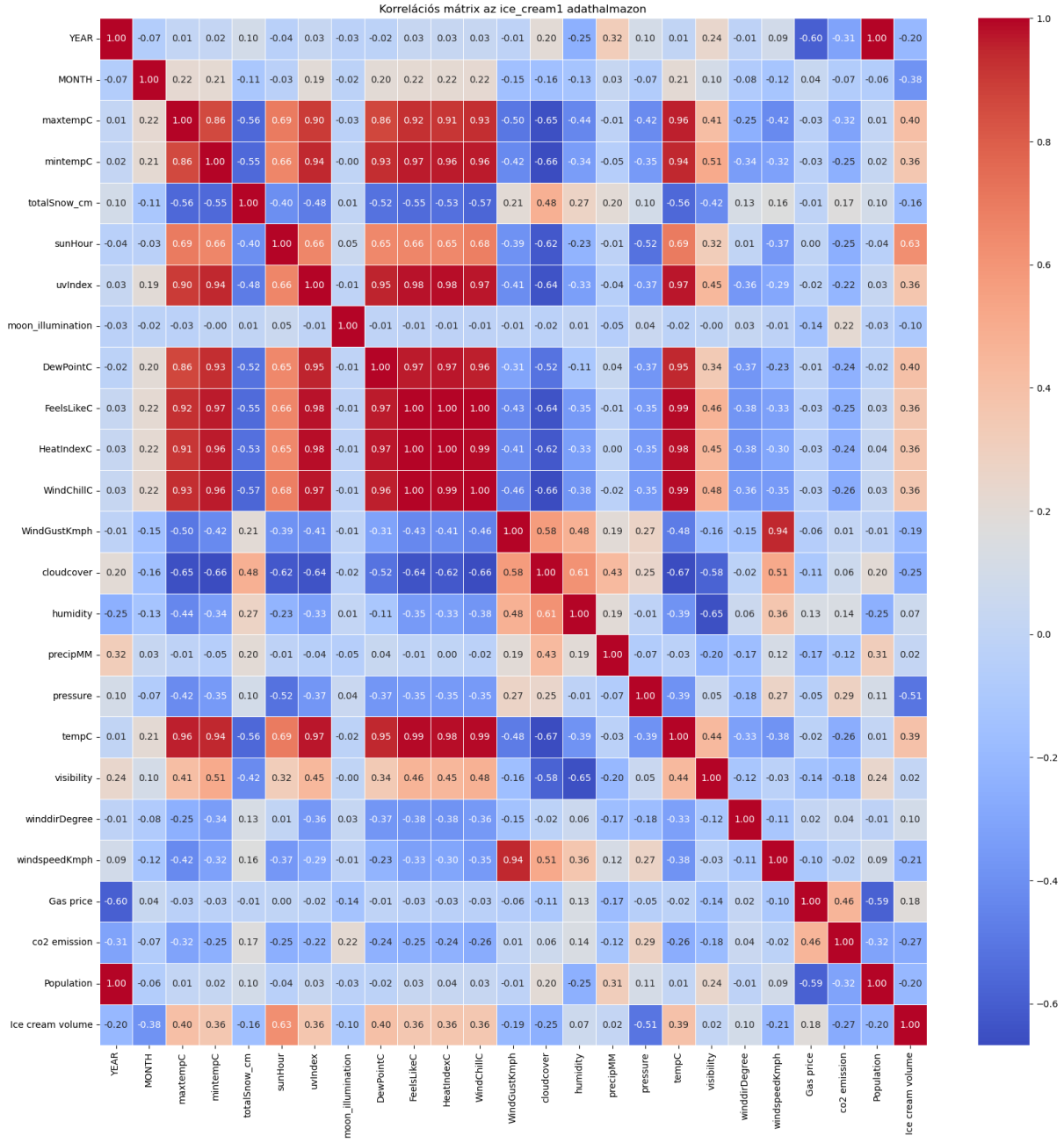
A(z) **uvIndex** és **Ice cream volume** közötti korreláció az ice_cream adathalmazban: **0.3621**

Heatmap – *weather1* (14. ábra) majd *ice_cream1* (15. ábra):

A 14. és 15. ábra a kiválasztott attribútumok egymáshoz képest viszonyított korrelációjukat mutatja be. Egy ilyen ábrával könnyedén tudjuk azonosítani az attribútumaink között fennálló lineáris kapcsolatok.



***14**. ábra – Az attribútumok közötti korrelációs együtthatók (napi bontás) – saját szerkesztés.*



***15**. ábra – Az attribútumok közötti korrelációs együtthatók (havi bontás) – saját szerkesztés.*

Az aggregálással elértük, hogy a lineáris kapcsolatok felerősödjenek.

4. Modellek felépítése, elméleti hátterük bemutatása:

4.1. Modellválasztás indoklása:

A modellek kiválasztásakor a cél az volt, hogy olyan módszereket alkalmazzak, amelyek képesek pontosan előre jelezni a fagylaltfogyasztást, figyelembe véve az időjárás, földgázárak, CO_2-kibocsátás és népesség hatásait. A lineáris és nem-lineáris regressziók jó alapot biztosítottak a független változók és a fogyasztás közötti kapcsolatok felmérésére. A lineáris regresszió egyszerű és jól érthető, de mivel az adatok gyakran nem lineáris kapcsolatokat tartalmaznak, szükség volt a nem-lineáris modellekre, mint például a neurális hálózatok és a Random Forest algoritmus. A neurális hálózatok különösen erősek a bonyolult, nem-lineáris mintázatok felismerésében, így pontosabb előrejelzéseket tettek lehetővé. A Random Forest jól kezeli a változók közötti interakciókat, miközben csökkenti a túlilleszkedés kockázatát, így stabil és pontos előrejelzéseket ad. Az idősoros jelleg miatt az ARIMA és LSTM modellek alkalmazása is indokolt volt. Az ARIMA a szezonális és időbeli függőségeket, míg az LSTM a hosszú távú időbeli trendeket és mintázatokat képes felismerni, így mindkét modell hozzájárul a jövőbeli fogyasztás pontosabb előrejelzéséhez. A választott modellek kombinációja tehát lehetővé teszi a fagylaltfogyasztás komplex összefüggéseinek és időbeli változásainak hatékony elemzését és előrejelzését.

4.2. Lineáris regressziós modell:

A lineáris regresszió [7] egy alapvető statisztikai és gépi tanulási módszer, amely lehetővé teszi a különböző változók közötti összefüggések modellezését. A lineáris modell célja, hogy a magyarázó változók (független változók) segítségével előrejelezze a célváltozót (függő változót). A lineáris regresszió egyszerűsített formája azt feltételezi, hogy a célváltozó és a magyarázó változók közötti kapcsolat egyenes, azaz lineáris. A modell a következő egyenlet alapján működik:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

Ahol:

- y a célváltozó
- β_0 az intercept vagy konstans, amely a magyarázó változók nélküli alap értéket jelenti,
- $\beta_1,\beta_2,\ldots,\beta_n$ a különböző magyarázó változókhoz tartozó súlyok, amelyek azt mutatják meg, hogy egy-egy változó mennyire befolyásolja a célváltozót,

- $x_1, x_2, \ldots, x_n$ a független változók, mint például a hőmérséklet, földgázárak, CO_2-kibocsátás, népesség, stb.,
- ϵ pedig a hiba, amely a modell által nem magyarázott részt jelenti.

A modell paramétereinek értékeit a gépi tanulási algoritmus keresése határozza meg, azaz úgy állítja be a súlyokat ($\beta_1, \beta_2, \ldots, \beta_n$), hogy minimalizálja a tényleges és a becsült értékek közötti eltérést (azaz a hibát). A lineáris regresszió a legkisebb négyzetek módszerét használja, hogy ezt a minimális eltérést elérje.

4.3. Neurális hálózat: MLPRegressor

A **MLPRegressor (Multi-layer Perceptron Regressor)** [8] egy neurális hálózaton alapuló regressziós modell, amelyet a scikit-learn könyvtár biztosít. A modell egy többrétegű perceptron (MLP) architektúrát használ, amely több rejtett rétegből és egy kimeneti rétegből áll. Az MLPRegressor az alapvető lineáris modellekkel szemben képes a nem-lineáris összefüggések és komplex mintázatok felismerésére, így ideális választás olyan esetekben, ahol a kapcsolat a bemeneti változók és a célváltozó között nem egyszerű, lineáris jellegű. A modell működésének alapja, hogy a bemeneti adatokat átadja a rejtett rétegeken, ahol súlyok és aktivációs függvények segítségével a modellezés során megtanulja a bemenetek és a kimenet közötti kapcsolatot.

A **MLPRegressor** modell paraméterei közül a legfontosabbak a **rétegméretek** (hidden_layer_sizes), amely meghatározza a rejtett rétegek számát és azok méretét, valamint a **max_iter**, ami az iterációk számát adja meg, vagyis hogy hányszor történjenek meg az optimalizációs lépések a tanulás során. Ezen kívül az **activation** paraméter meghatározza az aktivációs függvényt, amely lehet például **relu** (Rectified Linear Unit), **tanh**, vagy **logistic**. A tanulási folyamat során a súlyokat a **backpropagation** algoritmus segítségével optimalizálja, mely során a hiba visszaterjedésével finomítja a hálózat súlyait. A modell folyamatosan finomítja a bemeneti és kimeneti változók közötti kapcsolatot, hogy minél pontosabban előrejelezze a célváltozó értékét.

A **MLPRegressor** által használt aktivációs függvények és több rétegű architektúrák lehetővé teszik a komplex, nem-lineáris összefüggések modellezését, így jól alkalmazható olyan problémákra, ahol a bemeneti változók közötti kölcsönhatások nem lineárisak, mint például az időjárási, gazdasági adatok és a fogyasztási szokások közötti összefüggés. A **hidden_layer_sizes**

paraméter megfelelő beállításával a modell képes mélyebb összefüggéseket is felismerni, szemben egy egyszerű lineáris vagy döntési fa alapú modell képességeivel.

4.4 Random Forest

A **Random Forest** [6] egy rendkívül erőteljes és széles körben alkalmazott gépi tanulási algoritmus, amely döntési fákat használ egyesített (ensemble) modellként. Az algoritmus célja, hogy a döntési fák előnyeit összegyűjtse, miközben minimalizálja a modell túlilleszkedését (overfitting) és javítja az előrejelzés pontosságát. A Random Forestet mind osztályozási, mind regressziós feladatokhoz alkalmazzák, és különösen jól teljesít olyan esetekben, amikor az adatok komplexek és a változók közötti kapcsolatok nem lineárisak.

A Random Forest működése az **ensemble learning** (halmazati tanulás) elvén alapul, amely több egyszerű modell kombinálásával egy erősebb, robusztusabb előrejelzést eredményez. Az algoritmus több döntési fát hoz létre, amelyek mindegyike egy-egy különböző véletlenszerű mintán tanul, amit **bootstrap módszernek** neveznek. Ennek eredményeként minden egyes fa különböző mintákkal dolgozik, így a modellek eltérő döntéseket hoznak, ami növeli a stabilitást és csökkenti a varianciát. A fákat úgy építik fel, hogy minden egyes döntési elágazásnál véletlenszerűen választanak ki egy kisebb részhalmazt a változók közül, így biztosítva a modellek közötti diverzitást.

A Random Forest előnyei közé tartozik, hogy képes kezelni a hiányzó adatokat, nem érzékeny a zajos adatokra, és jól alkalmazható nagy, komplex adatállományokkal. Az egyes fák eredményeit aggregálják, hogy a végső döntést meghozzák: osztályozás esetén többségi szavazással, míg regresszió esetén az átlagolás alapján. Ez a megközelítés biztosítja a modell robusztusságát és csökkenti az overfitting kockázatát.

A Random Forest alkalmazási területe rendkívül széleskörű. Különösen jól használható a pénzügyi szektorban, például hitelminősítésben vagy pénzügyi előrejelzésekben, az egészségügyben betegségek előrejelzésére és diagnózisok osztályozására, valamint a marketingben a vásárlói magatartás előrejelzésére. Az algoritmus emellett alkalmas a változók fontosságának mérésére, ami segíthet az attribútumok szelektálásában és a modell finomhangolásában.

Bár a Random Forestnek megvannak a maga előnyei, nem mentes néhány hátránytól sem. Az algoritmus számítási igénye és memóriahasználata jelentős lehet, különösen nagy adathalmazok esetén, mivel sok fát kell létrehozni és tárolni. Ezen kívül a modellek értelmezhetősége

korlátozott, mivel a Random Forest egy összetett modell, amely több döntési fa eredményeit aggregálja. Kisebb attribútum szám esetén könnyedén reprezentálható a működése, de a változók számának növelésével ez egyre bonyolultabbá, és kevésbé értelmezhetővé válik. Ennek ellenére a Random Forest mégis egy kiemelkedően erőteljes eszköz, amely különösen jól alkalmazható olyan problémák megoldásában, ahol a klasszikus modellek nem biztosítanak megfelelő pontosságot vagy robusztusságot.

4.5. ARIMA:

Az ARIMA (**A**uto**R**egressive **I**ntegrated **M**oving **A**verage) [9] modell egy széles körben használt idősoros előrejelzési eszköz, amely a pénzügyi és időjárási elemzésekben kiemelkedő szerepet játszik. Az ARIMA modell célja, hogy az időbeli adatok előrejelzésére szolgáló prediktív modellt alkosson, figyelembe véve a múltbeli értékeket (AR), az előző idősorok értékeinek hatását (I) és az esetleges trendeket, ciklusokat (MA). Az ARIMA három fő összetevőből épül fel: az autoregresszív (AR) komponensből, a mozgóátlag (MA) komponensből, és az integrált (I) komponensből, melyek külön-külön és együttesen segítenek a legjobb előrejelzés elérésében.

Az **autoregresszív (AR)** komponens a múltbeli értékek lineáris kombinációjaként próbálja meg modellezni a jövőbeli értékeket. Az AR rész azt jelzi, hogy az adott idősor értékeinek előrejelzéséhez figyelembe kell venni a múltbeli megfigyeléseket. Az **integrált (I)** komponens az idősor differenciálását jelenti, amelyet azért alkalmazunk, hogy eltávolítsuk a nem állandó (stacionárius) trendet, és a változók stabilizálásával elérjük, hogy az idősor folytatható legyen előrejelzés céljából. A **mozgóátlag (MA)** komponens az idősor korábbi hibáinak lineáris kombinációjaként működik, amely segít modellezni azokat az értékeket, amelyek nem szerepelnek az autoregresszív vagy integrált komponensekben. Az ARIMA modell tehát három fő paraméterből áll: p (az autoregresszív komponens lagjainak száma), d (az idősor differenciálásának mértéke), és q (a mozgóátlag komponens lagjainak száma).

Az ARIMA modell erőssége, hogy képes megbirkózni az idősorok különböző típusú trendjeivel és ciklikus mintáival, miközben viszonylag egyszerű és könnyen alkalmazható eszköz. Azonban az modell használata előtt szükséges, hogy az idősor stacionárius legyen, azaz a statisztikai jellemzők (például az átlag és a szórás) ne változzanak az időben. Ha az idősor nem stacionárius, akkor a differenciálás vagy egyéb előfeldolgozási lépések szükségesek ahhoz, hogy a modell alkalmazható legyen.

Az ARIMA modell nagy előnye, hogy jól alkalmazható olyan időbeli adatokra, amelyekben a múltbeli megfigyelések meghatározóak a jövőbeli értékek előrejelzésében, de nem feltétlenül rendelkeznek bonyolult szezonalitással vagy más komplex dinamikákkal. Az **ARIMA** kiterjesztése a SARIMA (Seasonal ARIMA), amely a szezonális hatásokat is figyelembe veszi így tehát képes az idősoros adatok szezonális mintázatait jobban kezelni.

A modell alkalmazása során fontos figyelembe venni az ARIMA paraméterek optimális beállítását, ami gyakran automatikus keresési technikákkal történik (pl. AIC vagy BIC értékek alapján). Az optimális paraméterek kiválasztása kulcsfontosságú a modell előrejelzésének a pontossága szempontjából.

Az ARIMA modell különösen jól alkalmazható gazdasági előrejelzésekhez, pénzügyi adatok elemzéséhez, keresleti előrejelzésekhez, vagy bármilyen olyan problémához, ahol az időbeli trendek és korábbi mintázatok meghatározóak a jövőbeli viselkedés előrejelzésében. Az ARIMA modell azonban nem biztosít megoldást olyan adatokra, ahol erős szezonális hatások vagy nem-lineáris minták dominálnak, ilyenkor érdemes más modelleket alkalmazni, például LSTM.

4.6. LSTM:

Az **LSTM** (Long Short-Term Memory) [10] [11] egy speciális típusú neurális hálózat, amelyet az idősoros előrejelzés és a hosszú távú függőségek modellezésére fejlesztettek ki. Az LSTM a hagyományos visszafelé terjedő hálózatok (RNN) továbbfejlesztett változata, amely képes megoldani azokat a problémákat, amelyek a hagyományos RNN-ek esetében előfordulnak, mint például a gradiens eltűnése és robbanása, amelyek megnehezítik a hosszú távú memória fenntartását. Az LSTM különleges felépítése révén alkalmas olyan problémák kezelésére, amelyek hosszú távú összefüggéseket igényelnek az adatokban, így kiválóan alkalmazható időbeli minták felismerésére és előrejelzésére.

Az LSTM architektúrája három fő komponenst tartalmaz: az **input gate**, a **forget gate** és az **output gate**. Ezek a komponensek lehetővé teszik az információ szabályozott átáramlását és megtartását a hálózaton belül. Az **input gate** felelős azért, hogy meghatározza, mely információk kerüljenek be a memória cellába a következő lépésben. A **forget gate** dönt arról, hogy a memória cella mely részei kerüljenek eltávolításra, így segítve a nem releváns információk

törlését. Az **output gate** pedig meghatározza, hogy a memória cellából mely információk kerülnek a következő időlépésbe, illetve a modell kimenetéhez.

Az LSTM modell fő előnye, hogy képes megőrizni a hosszú távú függőségeket, míg a hagyományos neurális hálózatok és RNN-ek hajlamosak a korábbi információk gyors elfelejtésére. Éppen ezért az LSTM különösen hasznos olyan feladatok esetén, amelyek hosszú távú időbeli mintákat, trendeket, függőségeket vagy ismétlődő ciklusokat tartalmaznak, mint például pénzügyi előrejelzések vagy időjárás-előrejelzések.

Az LSTM hátránya, hogy a modell komplexitása miatt több számítási erőforrást igényel, mint a hagyományos statisztikai modellek, és az edzéshez hosszabb időre van szükség, különösen nagy adathalmazok esetén. Ezen kívül a hiperparaméterek megfelelő beállítása, mint például a memóriarétegek száma kihívást jelenthet a modell optimalizálása során.

Összességében az LSTM egy rendkívül erőteljes eszköz, amely lehetővé teszi az idősoros adatok komplex és hosszú távú összefüggéseinek hatékony modellezését, így kiemelkedő szerepet játszik a gépi tanulás és az adatbányászat területén, különösen olyan alkalmazásoknál, amelyek dinamikusan változó, hosszú távú előrejelzéseket igényelnek.

4.7. Teljesítménymérő eszközök:

A következő teljesítménymutatókat használtuk az modellek kiértékeléséhez: az R^2, MSE, RMSE, és a MAE.[6] [12]

4.7.1. R^2 mutató – Determinációs Együttható:

Az R^2 mutató, vagy determinációs együttható azt méri, hogy a modell mennyire képes megmagyarázni a célváltozó varianciáját. Az értéke (elméletileg) 0 és 1 között mozog, ahol az 1 azt jelenti, hogy a modell az összes varianciát tökéletesen megmagyarázza, a 0 pedig azt jelenti, hogy a magyarázó változók nem magyarázzák a célváltozó varianciáját. Gyakorlatban viszont előfordulhat, hogy a determinációs együttható negatív értéket vesz fel, ami azt jelenti, hogy a modellünknél még egy egyszerű átlag becslés is kevesebb hibával dolgozik. Az R^2 mutató kiszámítása az alábbi képlettel történik:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

ahol:

- y_i az i-edik megfigyelés valós (valódi) értéke,

- $\hat{y}_i$ az *i*-edik megfigyelés modell által adott előrejelzése,
- $\bar{y}$ az függő változó átlaga.

Az R^2 pontszám gyakran használt indikátor a regressziós modellek értékelésére, mivel jól szemlélteti, hogy a modell mennyire képes megragadni a változók közötti kapcsolatokat.

4.7.2. Mean Squared Error (MSE):

Az MSE (Mean Squared Error) a modell előrejelzései és a tényleges célértékek közötti négyzetes eltérések mértékét adja meg. Az MSE kiszámítása során a hibákat – azaz az előrejelzett és valódi értékek különbségeit – négyzetre emeljük, majd átlagoljuk, így nagyobb súlyt kapnak a jelentősebb hibák. Az MSE ezért különösen érzékeny a kiugró értékekre, amelyek torzíthatják a modell teljesítményének becslését, amennyibe azok nagy eltéréseket mutatnak. Az MSE-t az alábbi képlettel számítják:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

ahol:

- n a megfigyelések száma,
- y_i az *i*-edik megfigyelés valós (valódi) értéke,
- $\hat{y}_i$ az *i*-edik megfigyelés modell által adott előrejelzése,

Az MSE mutatót úgy is értelmezhetjük, mint a hibák varianciája, mivel átlagosan méri az előrejelzési értékek eltérését a tényleges értékektől. Ez az érték jól szemlélteti, hogy a modell hibái mennyire változnak a valós értékek körül.

4.7.3. RMSE (Root Mean Squared Error):

Az RMSE (Root Mean Squared Error) az átlagos négyzetes hiba négyzetgyöke, és a modell előrejelzési hibáinak mértékét adja meg. Az RMSE a leggyakrabban használt mérőszámok egyike a regressziós modellek értékelésére, mivel jól kezeli a nagyobb hibákat is, hiszen azok a négyzetre emelés miatt nagyobb súlyt kapnak. Az RMSE kiszámítása a következő módon történik:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Ez a mutató érzékeny a kiugró értékekre, ami fontos tényező lehet olyan esetekben, ahol a kiugró értékek hibáinak a minimalizálása az egyik fő cél. Az RMSE mutatóra gondolhatunk úgy is, mint a hiba szórása.

4.7.4. MAE – Átlagos Abszolút Hiba (Mean Absolute Error):

A MAE (Mean Absolute Error) azt méri, hogy a modell előrejelzései mennyire térnek el átlagos abszolút értékben a tényleges célértékektől. A MAE kiszámítása során a hiba abszolút értékét vesszük, majd átlagoljuk, így a mérték kevésbé érzékeny a kiugró értékekre, mint az MSE vagy az RMSE. Az MAE képlete a következő:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Az MAE egyszerűen értelmezhető, mivel azt jelzi, hogy az előrejelzések átlagosan mekkora hibát mutatnak, ami segít a modell pontosságának általános megértésében.

4.8 Attribútumok kiválasztása:

A gépi tanulási modellek hatékonyságának és pontosságának növelése érdekében az egyik kulcsfontosságú lépés az attribútumkiválasztás. Ennek célja, hogy az adathalmazban található számos változó közül azokat emeljük ki, amelyek a legnagyobb hatással vannak a célváltozóra, miközben csökkentjük az irreleváns vagy zavaró (zajos) tényezők számát. Az sklearn.feature_selection.SelectKBest [6] [13] függvény egy hatékony eszköz erre a célra.

A SelectKBest egy egyszerű és hatékony módszer az attribútumok szűrésére. Az algoritmus az adatok jellemzőit külön-külön kiértékeli egy felhasználó által megadott statisztikai teszt segítségével, majd rangsorolja azokat. A kiválasztási folyamat során csak a legjobb **k** attribútumokat tartja meg, ahol **k** értéke szabadon beállítható. Ez a módszer különösen hasznos, ha a nagy adathalmazban számos olyan attribútum található, amelyek gyenge kapcsolatban vagy semmilyen kapcsolatban nincsenek a célváltozóval, és ezek csak zajt adnak hozzá a modellhez.

A SelectKBest az alábbi lépésekben működik:

1. **Statisztikai teszt kiválasztása:** A felhasználó kiválaszt egy statisztikai tesztet az attribútumok és a célváltozó közötti kapcsolat mérésére.
    - **Regressziós problémák esetén:** például az ’f_regression’ használható.

- Kategorizációs problémák esetén: az ’chi2’ vagy ’f_classif’ ajánlott.

2. **Értékelés:** A kiválasztott statisztikai teszt minden egyes attribútumot külön kiértékel, és egy pontszámot rendel hozzá.
3. **Attribútumok szűrése:** A legjobb **k** attribútumot tartja meg a pontszámok alapján.

Beállítások:

A **SelectKBest** legfontosabb paraméterei:

- **score_func**: A statisztikai teszt, amely az attribútumok kiértékelésére szolgál.
- **k**: Az attribútumok száma, amelyeket meg szeretnénk tartani. Ha például 10-et szeretnénk, az érték **k=10**.

Példa:

```
selector = SelectKBest(f_regression, k=10)
X_new = selector.fit_transform(X, y)
```

A mi esetünkben az *f_regression* [14] statisztikai tesztet fogjuk alkalmazni. Az *f_regression* az F tesztet használja a magyarázó változók kiértékelésekor, az alábbi két lépésben teszi meg mindezt:

1. Kereszt korrelációk kiszámítása: a függvény kiszámítja a magyarázó változók és a célváltozó közötti korrelációkat a következő képlet alapján:
$$r = \frac{E[(X[:,i] - mean(X[:,i])) * (y - mean(y))]}{(std(X[:,i]) * std(y))}$$
2. Az így kapott korrelációs értékeket *F-értékekké* alakítja, majd kiszámítja a hozzájuk tartozó *p-értékeket*.

Ezt követően a magyarázó változókat *F-érték* és *p-érték* alapján rendezi sorba, és visszaadja az **k** legerősebb lineáris kapcsolatot mutató attribútumot.

5. Modellek beállítása és értékelése:

A dolgozatban összesen négy adathalmaz és öt különböző modell kerül felhasználásra. Az előrejelzési feladat megoldása tehát húsz különböző modell fejlesztését jelenti. Az 5. fejezetben minden egyes modell esetében a legjobban teljesítő változat kerül bemutatásra, amelyek az adott modell szempontjából a legpontosabb előrejelzéseket biztosították. A modellek kiértékelésekor bevezetem az „Általános hiba" mutatót mely az RMSE és a MAE átlaga.

5.1 Lineáris regresszió:

5.1.1 Adatok:

A lineáris regresszió modell a legjobb teljesítményt az ice_cream adattábla kiugró értékektől mentes verziójában érte el.

5.1.2 Modell létrehozása:

Az alábbi kódrészlet bemutatja a modell létrehozását:

```
# Változók beállítása
X = ice_cream_ken.drop(['Ice cream volume', 'CITY'], axis=1)
y = ice_cream_ken['Ice cream volume']
# Numerikus oszlopok skálázása
column_transformer = ColumnTransformer(
    transformers=[('num',  StandardScaler(),  X.select_dtypes(inc-
lude=['float64', 'int64']).columns),]
)
# Adatok felosztása
X_train = X[X['YEAR'] < 2018]
y_train = y[X['YEAR'] < 2018]
X_test = X[X['YEAR'] >= 2018]
y_test = y[X['YEAR'] >= 2018]
# Feature selection - kiválasztjuk a legjobb 13 attribútumot
selector = SelectKBest(f_regression, k=13)
# Pipeline létrehozása a lineáris regresszióhoz
pipeline_lr = Pipeline(steps=[
    ('preprocessor', column_transformer),
    ('selector', selector),
    ('model', LinearRegression())
```

```
])
# Modell tanítása és előrejelzés
pipeline_lr.fit(X_train, y_train)
y_pred = pipeline_lr.predict(X_test)
```

5.1.3 Teljesítménymutatók:

- R^2 együttható: 0.646713761467745
- RMSE: 8.973706362759852
- MAE: 7.0143980109117

Átlagos hiba: 7.9940

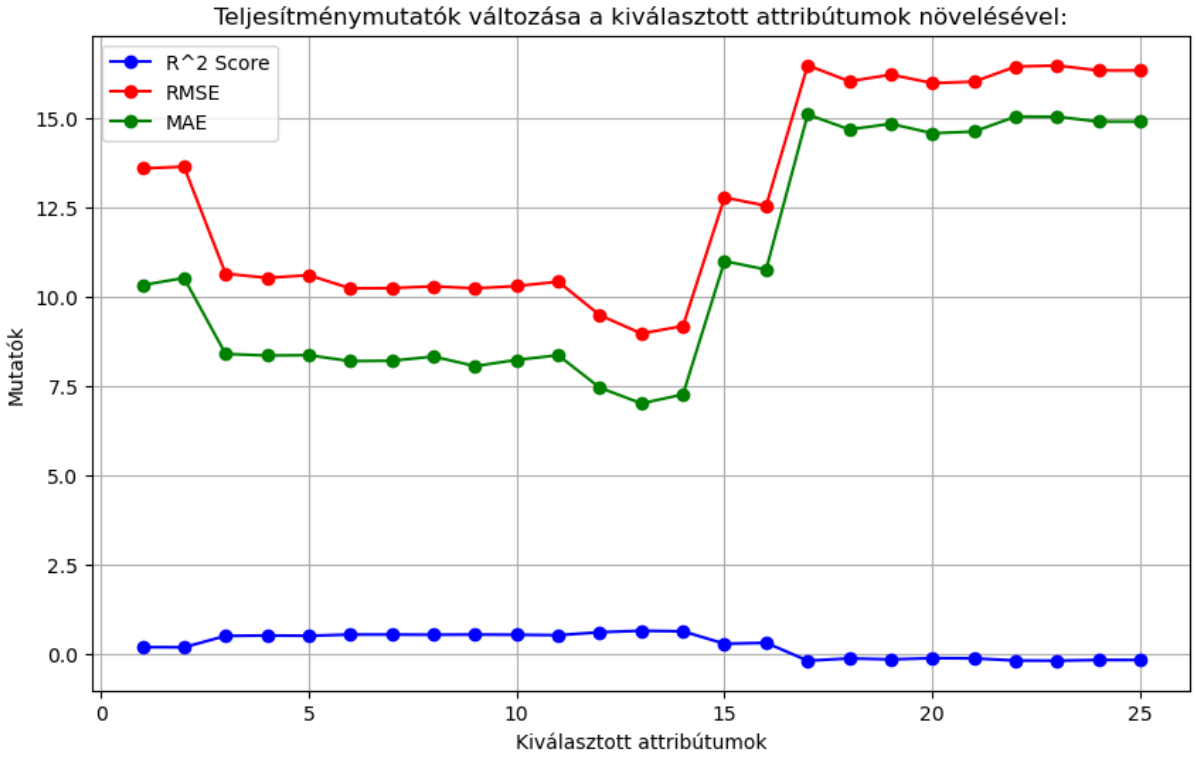
A lineáris regresszió a változó varianciájának a ~64,67%-át képes magyarázni, amely a felhasznált modellek közül a legalacsonyabb. Még a kiugró értékek kezelése ellenére is nehezen becsüli meg az adatoka. A modell általában 7.9940 millió gallon fagylaltfogyasztás téved.

Lineáris regressziós modell egyenlete:

$$y = 105.953 + (-7.008) * YEAR + (2.440) * MONTH + (3.059) * maxtempC + (9.548) * totalSnow\_cm + (-6.771) * sunHour + (10.711) * moon\_illumination + (15.297) * DewPointC + (10.467) * FeelsLikeC + (-47.029) * HeatIndexC + (-2.837) * precipMM + (11.401) * pressure + (-1.678) * winddirDegree + (-2.890) * Gas\ price$$

A legfontosabb változók között több hőmérséklettel kapcsolatos attribútum is szerepel, amelyek a korrelációs hőtérkép alapján multikollinearitást mutatnak. Ennek ellenére a redundánsnak tűnő információk eltávolítása jelentős teljesítménycsökkenést eredményezett a modellben. A modellek értékelése során az elsődleges szempontom a minél magasabb R^2 mutató elérése volt.

Az 16. ábra szemlélteti, hogy hogyan változnak a teljesítménymutatók az attribútumok számának növelésével:



16. *ábra – A lineáris regressziós modell teljesítménymutatóinak változása a kiválasztott attribútumok növelésével – saját szerkesztés.*

5.1.4 Predikciók:

Valós értékek:	Becsült értékek:	Eltérés:
86.3506	89.826101	3.475501
104.8646	100.667329	- 4.197271
110.2590	106.911050	- 3.347950
118.9803	109.893671	- 9.086629
113.1661	123.015952	9.849852

3. *Táblázat – A valós értékek, a lineáris regressziós modell által becsült értékek és ezek közötti eltérések – saját szerkesztés.*

5.2 Neurális háló: MLPRegressor:

5.2.1 Adatok:

Az MLPRegressor a weather adattábla kiugró értékektől megtisztított változatán érte el a legjobb teljesítményt.

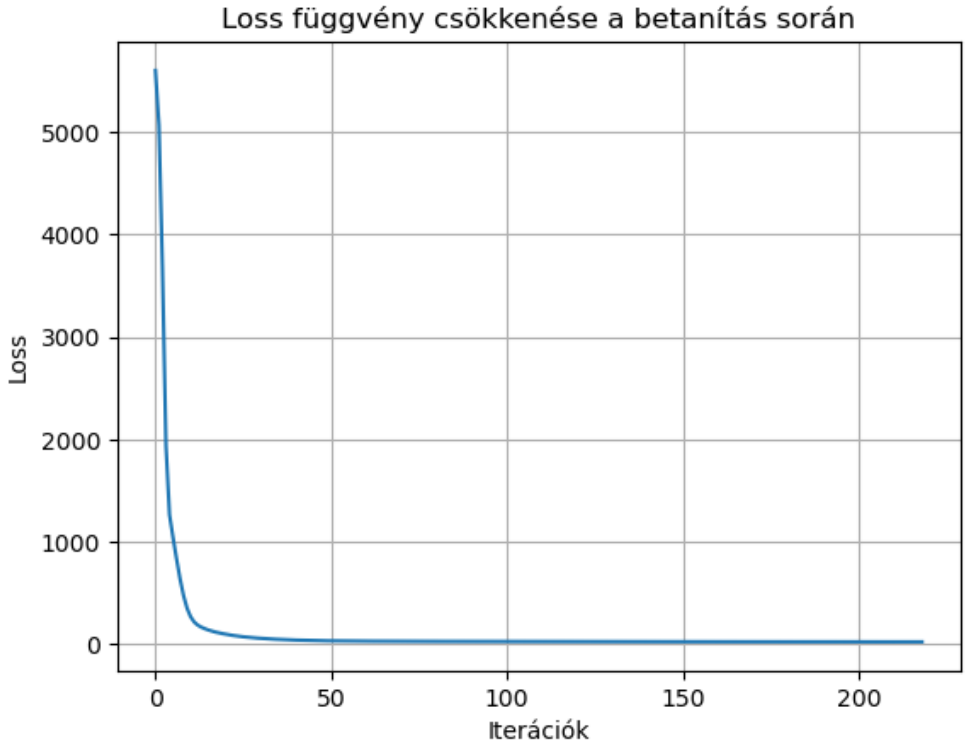
5.2.2 Modell létrehozása:

Az MLPRegressor neurális hálózat létrehozását az alábbi kódrészlet mutatja be:

```
# Célváltozó és attribútumok kiválasztása
X = weather_ken.drop(columns=['Ice cream volume', 'DATE'])
y = weather_ken['Ice cream volume']
# Train-test split az év alapján
X_train = X[X['YEAR'] < 2018]
y_train = y[X['YEAR'] < 2018]
X_test = X[X['YEAR'] >= 2018]
y_test = y[X['YEAR'] >= 2018]
# Numerikus és kategóriás oszlopok meghatározása
numeric_features = X.select_dtypes(include=[np.number]).columns.to-
list()
categorical_features = ['CITY']
# Előfeldolgozó pipeline létrehozása numerikus és kategóriás oszlo-
pokra
column_transformer = ColumnTransformer(
    transformers=[('num', StandardScaler(), numeric_features),
                  ('cat',   OneHotEncoder(handle_unknown='ignore'),
categorical_features)]
)
# Feature selection - kiválasztjuk a legjobb 12 attribútumot
selector = SelectKBest(f_regression, k=12)
# Pipeline létrehozása a neurális hálóhoz
pipeline_nn = Pipeline(steps=[
    ('preprocessor', column_transformer),
    ('selector', selector),  # Feature selection lépés
    ('model',              MLPRegressor(hidden_layer_sizes=(50,12),
max_iter=750, random_state=42, early_stopping=True))
```

```
])
# Modell betanítása és predikció készítése:
pipeline_nn.fit(X_train, y_train)
y_pred = pipeline_nn.predict(X_test)
```

A 17. ábrán láthatjuk a betanítás során a loss függvény (MSE) alakulását:



17. ábra – A MLPRegressor loss függvényének a csökkenése a betanítás során – saját szerkesztés.

5.2.3 Teljesítménymutatók:

- R^2 együttható: 0.8966644674820456
- RMSE: 4.966502125060943
- MAE: 4.093194858187541

Átlagos hiba: 4.5298

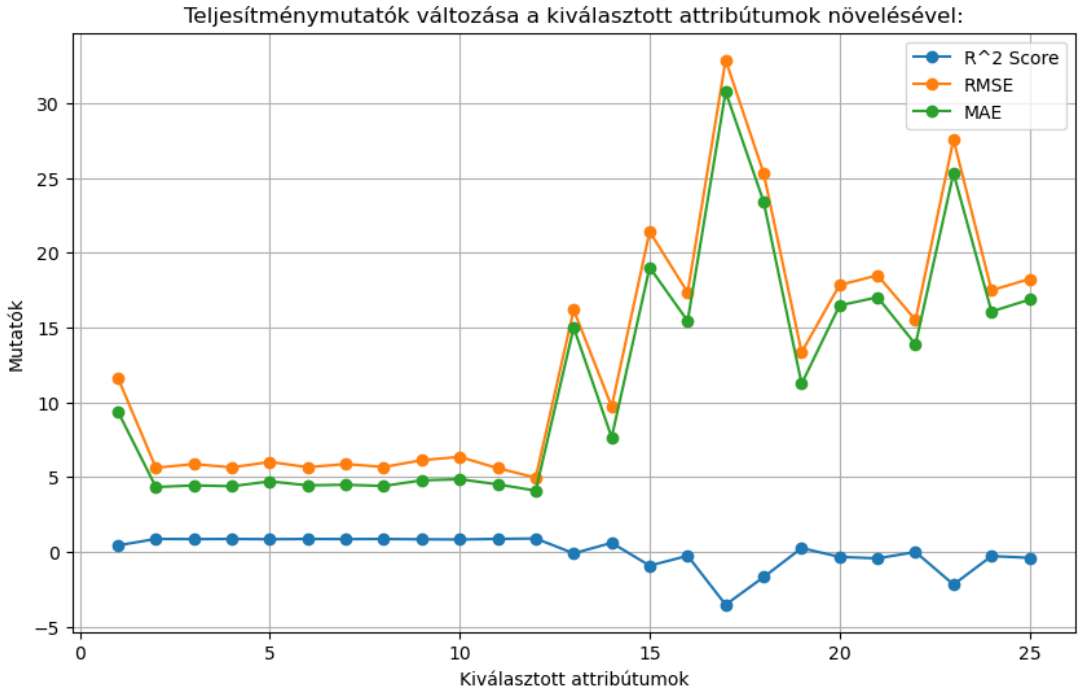
A MLPRegressor kiváló értékeket ért el, a célváltozó varianciájának 89,66%-át képes magyarázni, ami a weather adattáblán elvégzett modellek közül a legmagasabb. Átlagos hibáját tekintve második helyen áll a weather adattáblában. A modell általában 4.5298 millió gallon fagylaltfogyasztás téved.

A neurális háló a becslések elvégzéséhez az alábbi attribútumokat használta fel:

Attribútum	**F próba**	**p-érték**
sunHour:	F-próbastatisztika = 7747.00	p-érték = 0.00e+00
MONTH:	F-próbastatisztika = 3558.01	p-érték = 0.00e+00
maxtempC:	F-próbastatisztika = 1752.59	p-érték = 0.00e+00
tempC:	F-próbastatisztika = 1752.59	p-érték = 0.00e+00
DewPointC:	F-próbastatisztika = 1455.18	p-érték = 7.82e-303
HeatIndexC:	F-próbastatisztika = 1371.29	p-érték = 2.71e-286
uvIndex:	F-próbastatisztika = 1351.14	p-érték = 2.63e-282
FeelsLikeC:	F-próbastatisztika = 1337.68	p-érték = 1.22e-279
WindChillC:	F-próbastatisztika = 1336.35	p-érték = 2.24e-279
pressure:	F-próbastatisztika = 1249.61	p-érték = 3.88e-262
co2 emission:	F-próbastatisztika = 1086.24	p-érték = 2.14e-229
mintempC:	F-próbastatisztika = 846.24	p-érték = 1.24e-180

Ezeket az attribútumokat a *SelectKBest(f_regression, k=12)* függvény válogatta ki számomra amely a célváltozóval **k** legerősebb lineáris kapcsolatot mutató attribútumokat választja ki.

A 18. ábra bemutatja, hogy az attribútum számának növelésével hogyan alakulnak a modell teljesítmény mutatói:



18. ábra – Az MLPRegressor teljesítménymutatóinak az alakulása a kiválasztott attribútumok számának növelésével – saját szerkesztés.

5.2.4 Predikciók:

Valós értékek:	Becsült értékek:	Eltérés:
86.3506	93.500840	7.150240
86.3506	92.079052	5.728452
86.3506	87.764464	1.413864
86.3506	88.790087	2.439487
86.3506	87.868261	1.517661

4. *Táblázat – A valós értékek, az MLPRegressor által becsült értékek és ezek közötti eltérések – saját szerkesztés.*

5.3 Random Forest:

5.3.1 Adatok:

A Random Forest algoritmus a weather adattábla kiugró értékektől mentes változatában teljesített a legjobban.

5.3.2 Modell létrehozása:

Az alábbi kódrészlet bemutatja a Random Forest létrehozását:

```
# Célváltozó és attribútumok kiválasztása
X = weather_ken.drop(columns=['Ice cream volume', 'DATE'])
y = weather_ken['Ice cream volume']
# Train-test split az év alapján
X_train = X[X['YEAR'] < 2018]
y_train = y[X['YEAR'] < 2018]
X_test = X[X['YEAR'] >= 2018]
y_test = y[X['YEAR'] >= 2018]
# Előfeldolgozó pipeline létrehozása numerikus és kategóriás oszlopokra
column_transformer = ColumnTransformer(
    transformers=[('num', StandardScaler(), X.select_dtypes(include=['float64', 'int64']).columns),
```

```
                    ('cat',   OneHotEncoder(handle_unknown='ignore'),
['CITY'])]
)
# Feature selection beállítása az aktuális k érték szerint (2 legjobb
attribútum kiválasztása)
selector = SelectKBest(f_regression, k=2)
# Pipeline létrehozása a random foresthez
pipeline_rf = Pipeline(steps=[
    ('preprocessor', column_transformer),
    ('selector', selector),
    ('model',      RandomForestRegressor(n_estimators=200,      ran-
dom_state=42, n_jobs=-1))
])
# Modell betanítása
pipeline_rf.fit(X_train, y_train)
# Előrejelzések készítése
y_pred = pipeline_rf.predict(X_test)
```

5.3.3 Teljesítménymutatók:

- R^2 együttható: 0.8937
- RMSE: 5.0364
- MAE: 3.8915

Átlagos hiba:4.4639

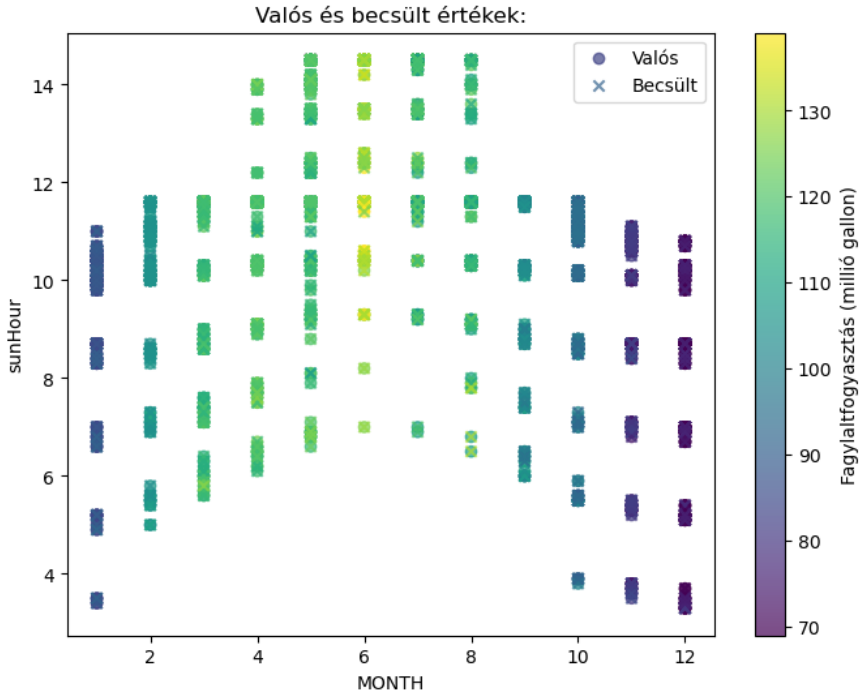
A weather adattáblán a Random Forest algoritmus a második legjobb eredményeket hozta, ha a determinációs együttható alapján döntünk, viszont átlagos hiba alapján a legjobb modell. A célváltozó szórásának a 89,37%-át képes magyarázni, átlagosan 4,4639 millió gallon fagylalt-fogyasztást téved.

A modell a következő attribútumokat használta fel:

Attribútumok fontosságai:

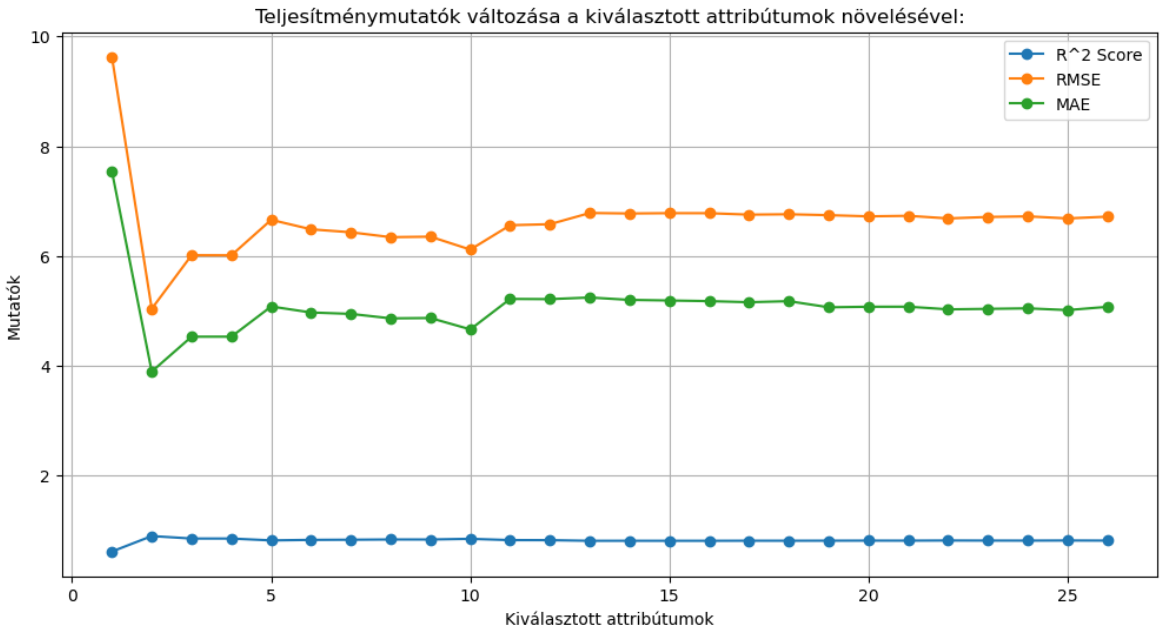
- MONTH: 0.9917
- sunHour: 0.0083

Ha komplexitását tekintjük ez a legjobb regressziós modell, amely csupán 2 attribútum mentén magyarázza a célváltozónkat. Az eredmények összhangban vannak a hipotézisemmel, hogy a fagylaltfogyasztás erősen szezonális jellegű. A két attribútum szerint a következőképpen becsülte meg a fogyasztást, a 19. ábrán látható módon:



19. ábra – A Random Forest becslései a napsütéses órák száma és hónapok alapján – saját szerkesztés

A teljesítménymutatók alakulása a kiválasztott attribútumok számának növelésével: (20. ábra)



20. ábra – A Random Forest teljesítménymutatóinak a változása a kiválasztott attribútumok számának növelésével – saját szerkesztés.

5.3.4 Predikciók:

Valós értékek:	Becsült értékek:	Eltérés:
86.3506	90.828759	7.150240
86.3506	87.094064	5.728452
86.3506	87.094064	1.413864
86.3506	87.094064	2.439487
86.3506	87.094064	1.517661

5. *Táblázat – A valós értékek, a Random Forest által becsült értékek és ezek közötti eltérések – saját szerkesztés.*

5.4 ARIMA:

5.4.1 Adatok:

Az ARIMA idősoros modell az ice_cream adattábla kiugró értékekkel rendelkező verziójában érte el a legjobb eredményeket.

5.4.2 Modell létrehozása:

Az alábbi kódrészlet bemutatja az ARIMA modell létrehozását:

```
#Adatok előkészítése:
arima_data_cities = ice_cream.copy()
data = arima_data_cities[arima_data_cities['CITY']=='Chicago'].copy()
data["DATE"] = pd.to_datetime(data["DATE"])  # Ha még nem lenne da-
tetime formátumban
data.set_index("DATE", inplace=True)
data = data.sort_index()
p, d, q = 9, 0, 9
model = ARIMA(data["Ice cream volume"], order=(p, d, q))
model_fit = model.fit()
model_summary = model_fit.summary()
print(model_summary)
# Train - test adatok felosztása:
train_size = int(len(data) * 0.8)
```

```
train, test = data["Ice cream volume"][:train_size], data["Ice cream volume"][train_size:]
# Modell betanítása a training adatokkal:
model = ARIMA(train, order=(p, d, q))
model_fit = model.fit()
# Predikciók készítése:
forecast = model_fit.forecast(steps=len(test))
```

5.4.3 Teljesítménymutatók:

- R^2 együttható: 0.9047
- RMSE: 4.9964
- MAE: 3.8239

Átlagos hiba: 4.4102

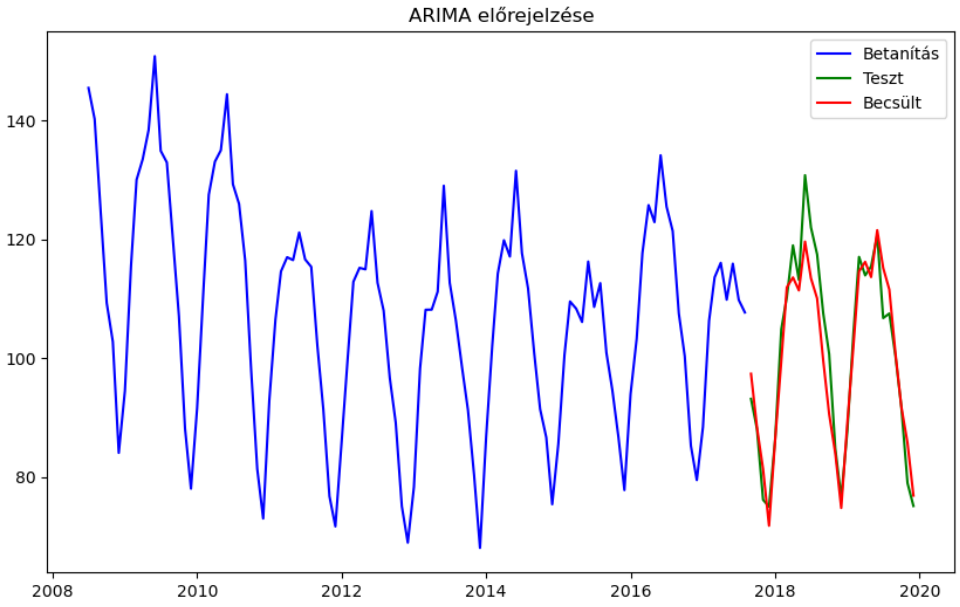
Az ice_cream adattáblán a havi adatokból az ARIMA idősoros modell a célváltozó szórásának 90,47%-át képes magyarázni, amely az összes modell közül a második helyet foglalja el determinációs együttható tekintetében. Átlagosan 4.4102 millió gallon fagylaltfogyasztást téved, amivel az összes modell közül a második legkevesebb hibával becsül.

5.4.4 Predikciók:

Valós értékek:	Becsült értékek:	Eltérés:
93.1503	97.377108	4.226808
88.0984	88.349986	0.251586
76.1579	81.312248	5.154348
75.0005	71.799487	- 3.201013
86.3506	86.279327	- 0.071273

6. *Táblázat – A valós értékek, az ARIMA modell által becsült értékek és ezek közötti eltérések – saját szerkesztés.*

A 21. ábra bemutatja a prediktált és valós értékeket:



21. ábra – Az ARIMA becslései, valamint a betanító és teszt adatok – saját szerkesztés.

5.5 LSTM:

5.5.1 Adatok:

Az LSTM modell az ice_cream adattáblának a kiugró értékekkel rendelkező változatában érte el a legjobb eredményeket.

5.5.2 Modell létrehozása:

Az alábbi kódrészlet szemlélteti a modell létrehozását:

```
# 1. Adatok előkészítése és tisztítása:
lstm_data_cities = weather.copy()
data2 = lstm_data_cities[lstm_data_cities['CITY']=='Chicago'].copy()
del data2['CITY']
data2['DATE'] = pd.to_datetime(data2['DATE'])
data2 = data2.sort_values('DATE')
# Célváltozó kiválasztása:
target_column = data2.columns[-1]  # Az utolsó oszlop a célváltozó
# 2. Normalizálás:
scaler = MinMaxScaler(feature_range=(0, 1))  # 0 és 1 közötti skálázás
numerical_columns = data2.columns[1:-1]  # Az első oszlop (DATE) és az
utolsó (célváltozó) kivételével
```

```
# Skálázás
data_scaled = scaler.fit_transform(data2[numerical_columns])
# 3. Idősoros bemeneti adatok előkészítése:
def create_dataset(data, target, time_step=10):
    X, y = [], []
    for i in range(len(data) - time_step - 1):
        X.append(data[i:(i + time_step), :])# Az első `time_step` napi
adatok
        y.append(target[i + time_step]) # A következő érték (célváltozó)
    return np.array(X), np.array(y)
target_column = 'Ice cream volume'
X,   y   =   create_dataset(data_scaled,   data2[target_column].values,
time_step=10)
# 4. Adatok szétválasztása tanító- és teszthalmazokra:
train_size = int(len(X) * 0.8)  # 80% tanító adat
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]
# 5. Bemeneti adatok átalakítása az LSTM-hez:
X_train    =    X_train.reshape(X_train.shape[0],    X_train.shape[1],
X_train.shape[2])
X_test     =     X_test.reshape(X_test.shape[0],     X_test.shape[1],
X_test.shape[2])
# 6. LSTM modell létrehozása:
model = Sequential()
# 1. LSTM réteg (Input gate):
model.add(LSTM(units=100,          return_sequences=True,          in-
put_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))  # Dropout réteg, hogy elkerüljük a túltanulást
(Overfitting)
# 2. LSTM réteg (Forget gate):
model.add(LSTM(units=50, return_sequences=False))
model.add(Dropout(0.2))  # Dropout réteg ismét a túltanulás elkerülése
érdekében
# 3. Kimeneti réteg (Output gate):
model.add(Dense(units=1))  # Csak egy célváltozó
```

```
# Modell összeállítása:
model.compile(optimizer='adam', loss='mean_squared_error')
# 7. Modellek betanítása és history mentése:
history = model.fit(X_train, y_train, epochs=3000, batch_size=16, vali-
dation_data=(X_test, y_test))
# 9. Előrejelzés és kiértékelés:
predicted_y = model.predict(X_test)
```

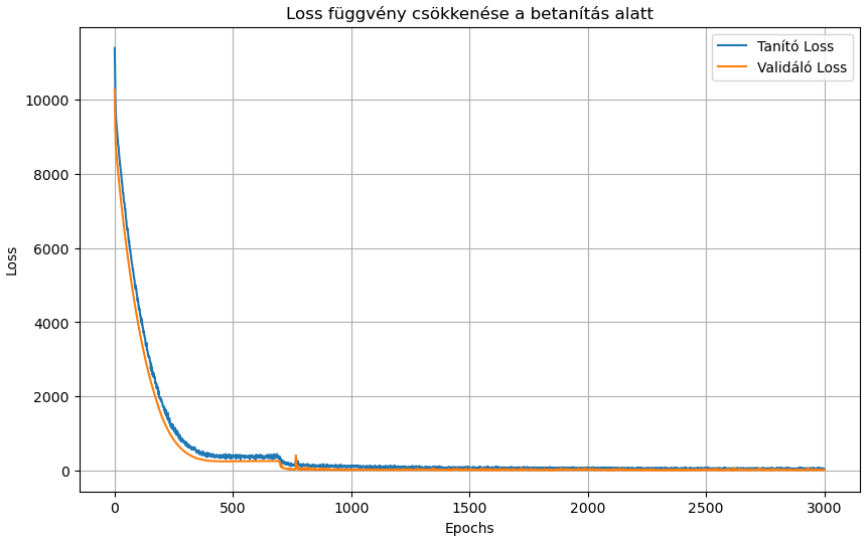
5.5.3 Teljesítménymutatók:

- R^2 együttható: 0.9173
- RMSE: 4.5683
- MAE: 3.8885

Átlagos hiba: 4.2284

Az ice_cream adattáblán a havi adatokból az LSTM idősoros modell a célváltozó szórásának 91,73%-át képes magyarázni, amely az összes modell közül az első helyet foglalja el determinációs együttható tekintetében. Átlagosan 4.2284 millió gallon fagylaltfogyasztást téved, amivel az összes modell közül az legkevesebb hibával becsül.

A modell betanítása közben a loss függvény (MSE) csökkenése a 22. ábrán látható:



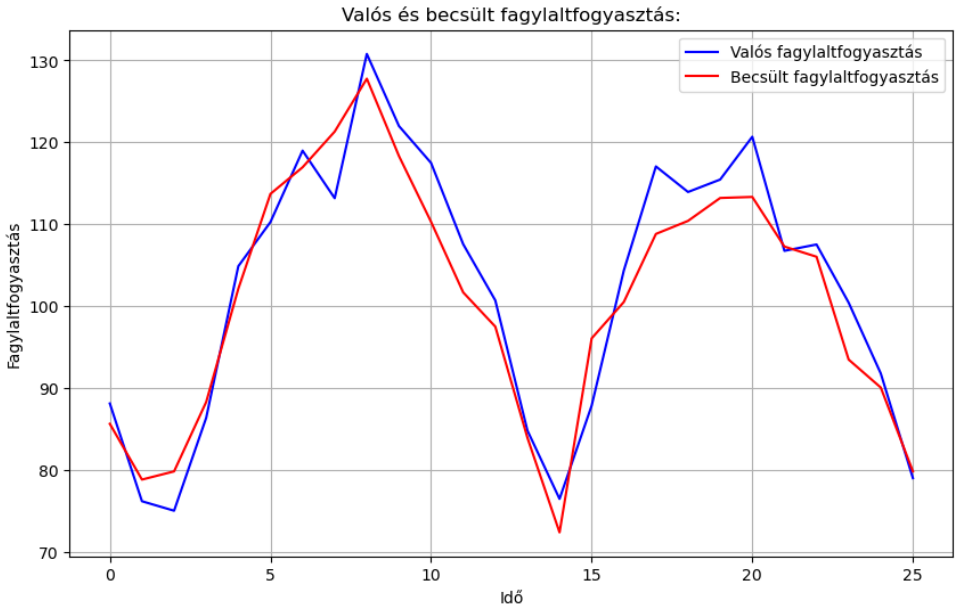
22. ábra – Az LSTM loss függvényének a csökkenése a betanítás során – saját szerkesztés.

5.5.4 Predikciók:

Valós értékek:	Becsült értékek:	Eltérés:
88.0984	85.614639	2.483761
76.1579	78.803505	2.645605
75.0005	79.803978	4.803478
86.3506	88.283546	1.932946
104.8646	102.111336	- 2.753264

7. Táblázat – A valós értékek, az LSTM modell által becsült értékek és ezek közötti eltérések – saját szerkesztés.

A 23. ábrán látható a valós és a prediktált értékek alakulása:



23. ábra – Az LSTM modell becslései a tesz adatokhoz viszonyítva – saját szerkesztés.

6. Eredmények értelmezése és megbeszélése:

A Weather1 adattábla alapján a legjobb modellek:

Modell:	Kiugró érték:	R^2:	RMSE:	MAE:	Átlagos hiba:
Lineáris Regresszió	Kezelve	0.6333	9.3558	7.5357	8.4457
MLPRegressor	Kezelve	0.8967	4.9665	4.0932	4.5298
Random Forest	Kezelve	0.8937	5.0364	3.8915	4.4639
ARIMA	Nem kezelve	0.4132	12.486	9.6664	11.076
LSTM	Nem kezelve	0.8873	5.4720	4.2257	4.8489

8. *Táblázat – A weather1 adathalmazon alkalmazott modellek legjobb teljesítménnyel rendelkező változatai – saját szerkesztés.*

Az Ice_cream1 adattábla alapján a legjobb modellek:

Modell:	Kiugró érték:	R^2:	RMSE:	MAE:	Átlagos hiba:
Lineáris Regresszió	Kezelve	0.6467	8.9737	7.0143	7.9940
MLPRegressor	Kezelve	0.8428	5.9853	4.9049	5.4451
Random Forest	Kezelve	0.8762	5.3119	4.2771	4.7945
ARIMA	Nem kezelve	0.9047	4.9964	3.8239	4.4102
LSTM	Nem kezelve	0.9173	4.5683	3.8885	4.2284

9. *Táblázat – Az ice_cream1 adathalmazon alkalmazott modellek legjobb teljesítménnyel rendelkező változatai – saját szerkesztés.*

A legjobb teljesítményt az *ice_cream1* adattáblán az LSTM rekurzív neurális hálóval értem el. Ennek oka, hogy az adatok előkészítése és természetük révén az LSTM modellhez illeszkedtek leginkább. Az LSTM architektúrája különösen jól kezelte az adatok dinamikáját és hosszú távú időbeli trendjeit, amelyeket más modellek nehezebben tudtak modellezni. Az LSTM R^2 mutatója 0.9173, míg átlagos hibája 4.2284 millió gallon.

Bár a Random Forest 2 attribútummal végzett becslései is figyelemre méltóak, mivel viszonylag alacsony komplexitás mellett is szép eredményeket produkált. R^2 mutatója 0.8937, átlagos hibája pedig 4.4639.

A modellek túlilleszkedésének vizsgálata:

A fagylaltfogyasztásának minimum értéke: 68.0609, maximum értéke: 150.8204, köztes esetben: 109,44065 *[(68.0609+150.8204)/2].* Ehhez viszonyítva a modellek átlagos hibái 11.076 és 4.2284 között alakultak, köztes esetben: 7,6522 *[(4.2284+11.076)/2]*

Összeségében:

- Legrosszabb esetben a hibák százalékos értéke: **16.27%** *[(11.076/68.0609)*100]*,
- Köztes esetben: **6.99%** *[(7,6522/109,44065)*100],*
- legjobb esetben: **2.80%** *[(4.2284/150.8204)*100].*

Az LSTM vonatkozásában:

- Legrosszabb esetben: **6.21%** *[(4.2284/68.0609)*100],*
- Köztes esetben: **3.86%** *[(4.2284/109,44065)*100],*
- Legjobb esetben **2.80%** *[(4.2284/150.8204)*100].*

Random Forest vonatkozásában:

- Legrosszabb esetben: **6.56%** *[(4.4639/68.0609)*100],*
- Köztes esetben: **4.07%** *[(4.4639/109,44065)*100],*
- Legjobb esetben **2.95%** *[(4. 4639/150.8204)*100].*

Ezek a számítások nem pontosak, ezért értelmezésük óvatos kezelést igényel. Csak extrém esetekben fordulhatnak elő, de segítenek meghatározni a hiba százalékos értékének alsó és felső határait.

Elképzelhető, hogy a modelljeink túlilleszkedtek az adatokra, ami azt jelenti, hogy túlzottan alkalmazkodtak a tanuló adathalmazhoz, és nem képesek jól általánosítani új, nem látott adatokra.

6.1. A kiválasztott attribútumok elemzése:

- A lineáris regresszióhoz kiválasztott attribútumok:
    - YEAR, MONTH, maxtempC, totalSnow_cm, sunHour, moon_illumination, DewPointC, FeelsLikeC, HeatIndexC, precipMM, pressure, winddirDegree, Gas price
- Az MLPRegressorhoz kiválasztott attribútumok:
    - sunHour, MONTH, maxtempC, tempC, DewPointC, HeatIndexC, uvIndex, FeelsLikeC, WindChillC, pressure, co2 emission, mintempC
- A Random Foresthez kiválasztott attribútumok:
    - MONTH, sunHour
- Az ARIMA modell csak a célváltozó alapján magyarázza a jövőbeli fogyasztást.
- Az LSTM modell minden attribútumot felhasznált.

Amint azt láthatjuk, a MONTH és a sunHour attribútumok mindegyik modellben jelen voltak, tehát kijelenthető, hogy a két attribútum nagy hatással van a fagylaltfogyasztásra. A SelectKBest függvény F-teszt hipotézisvizsgálatának eredményei alapján megállapítható, hogy az évszakok váltakozása jelentős hatással van a fagylaltfogyasztásra. A p-érték mindkettő attribútumnál magasabb, mint a szignifikancia szint, azaz a null hipotézist, miszerint a MONTH és a sunHour attribútumok nincsenek hatással a fagylaltfogyasztásra, elutasítjuk. Ez azt jelenti, hogy statisztikailag szignifikáns kapcsolat van az évszakok váltakozása, a napsütéses órák száma és a fagylaltfogyasztás között.

6.2. A modellek korlátai és kihívásai:

A lineáris regressziónak a legfőbb korlátja, hogy nem képes jól modellezni a nem lineáris kapcsolatokat. Az MLPRegressor érzékeny a hiperparaméterek beállítására, és nagyobb adathalmazok esetén költséges a betanítása. A Random Forest algoritmus túl sok döntési fa felállítása esetén és azok mélységének korlátozása hiányában olyan szabályokat is megtanulhat az adatokból, melyek a valóságban nincsenek jelen. Az ARIMA modell csak akkor képes jó előrejelzéseket készíteni, ha az adataink időben stacionáriusok. Az LSTM modell szintúgy érzékeny a hiperparaméterek beállítására és költséges lehet nagyobb adathalmazok betanítása esetén.

6.3. Továbbfejlesztési lehetőségek:

Nagyobb erőforrásokkal pontosabb adatgyűjtés végezhető, ami jelentősen hozzájárulhatna a modellek pontosságához. A napi fogyasztási adatok bevonása segíthetne jobban feltárni az attribútumok közötti kapcsolatokat. A változókiválasztás során észlelhető, hogy bizonyos változók között magas a korreláció, és több hőmérsékleti adatunk is egymásból számolható, így ezek a redundáns információk eltávolításával javíthatnánk a modellek teljesítményét és csökkenthetnénk azok komplexitását. A hiperparaméterek finomhangolásával, például automatizált keresési módszerekkel, mint a grid search, még jobb eredmények érhetők el.

Összefoglalás:

A szakdolgozatom célja olyan mélytanuló algoritmusokon alapuló modell kifejlesztése, amely képes pontosan előrejelezni a fagylaltfogyasztást. Ehhez különböző időjárási, környezeti és gazdasági adatokat használtam fel, mivel a fogyasztói magatartás előrejelzése elengedhetetlen a modern gyártástechnológiában és kereskedelemben, különösen olyan szezonális termékek esetén, mint a fagylalt, amelynek kereslete éves szinten jelentős ingadozásokat mutat.

A modellek fejlesztésében a Lineáris Regresszió, az MLPRegressor neurális hálózat, a Random Forest döntési fák, valamint az ARIMA és az LSTM idősoros modellek szerepeltek. A modellezést Python nyelven valósítottam meg, a Numpy, Pandas, Statsmodels, TensorFlow és Scikit-learn könyvtárak segítségével. A vizualizációk készítéséhez a Matplotlib és a Seaborn könyvtárakat alkalmaztam.

A kutatás során különböző forrásokból gyűjtött adatokat integráltam egy egységes adatbázisba. Az adatok között szerepeltek időjárási adatok, CO^2-kibocsátás, földgázárak, népességi adatok és a fagylaltfogyasztás. Az adatok tisztítása és előfeldolgozása komoly kihívásokat jelentett, mivel az adatok különböző terjedelműek és gyakoriságúak voltak, így számos hiányzó értéket kellett pótolni. A különböző modellek eltérő adatformátum-igényei is további feladatokat adtak.

Az elemzés során összesen öt modellt alkalmaztam négy különböző adathalmazon, hogy megtaláljam az optimális megoldásokat. Eredményeim azt mutatták, hogy a lineáris regresszió nem képes megfelelően előrejelezni a fagylaltfogyasztást, mivel a fogyasztói magatartás nem lineáris egyenlet mentén változik. Viszont a többi modell megfelelő adathalmazokkal és

hiperparaméter beállításokkal jól tudta magyarázni a célváltozót. A legjobb teljesítményt az idősoros modellek nyújtották, ami az adatok természetét figyelembe véve nem meglepő.

A kutatás eredményei azt mutatják, hogy a fogyasztási adatok egyszerűen és hatékonyan előrejelezhetők ezen a piacon. A legnagyobb hatással a fogyasztásra az időjárási tényezők voltak, mint az évszakok váltakozása és a napsütéses órák száma. A CO^2-kibocsátás és a földgázárak hatása viszont csak alig volt kimutatható. Az elkészült modellek alacsony hibával képesek előrejelezni a fagylaltfogyasztást, ami lehetőséget biztosít a termelési kapacitások optimalizálására, valamint fenntarthatósági szempontból is hasznos információkat nyújt, mivel segíthet elkerülni a felesleges készletek felhalmozását.

Bár a modellek jól teljesítenek, még van lehetőség a finomhangolásukra. Az automatizált hiperparaméter-keresés, a megfelelő gyakoriság szerinti adatgyűjtés, valamint az attribútumok közötti korreláció csökkentése mind hozzájárulhatnak a modellek pontosságának javításához. Emellett más modellek, például az ARIMA szezonális változata is alkalmazható lenne a jövőbeli kutatásokban.

Irodalomjegyzék:

Adathalmazok:

[1] https://www.kaggle.com/datasets/luisvivas/weather-north-america letöltés ideje: 2023.11.11.

[2] https://datahub.io/core/natural-gas letöltés ideje: 2023.11.10.

[3] https://www.eia.gov/totalenergy/data/browser/xls.php?tbl=T11.01 letöltés ideje: 2024.10.28.

[4] https://www.worldometers.info/world-population/us-population/ letöltés ideje: 2023.11.11.

[5] https://fred.stlouisfed.org/series/IPN31152N letöltés ideje: 2024.10.22.

Magyar:

[6] Mueallert, J. P., Massaront, L.: Adatelemzés Pythonnal. *Taramix kiadó kft.* 2024

Külföldi:

[7] Scikit learn: Lineáris regresszió dokumentációja: (https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LinearRegression.html), utoljára megtekintve: 2024.11.19.

[8] Dr. Powers, S. MLPRegressor: (https://www.youtube.com/watch?v=NpcdJyxv_3w) , utoljára megtekintve: 2024.11.19.

[9] Saadeddin, Z., ARIMA for Time Series Forecasting: A Complete Guide:(https://www.datacamp.com/tutorial/arima?dc_referrer=https%3A%2F%2Fwww.google.com%2F) , utoljára megtekintve: 2024.11.19.

[10] Banoula, M. Introduction to Long Short-Term Memory(LSTM): (https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/lstm) , utoljára megtekintve: 2024.11.19.

[11] StatQuest, Long Short-Term Memory (LSTM), Clearly Explained: (https://www.youtube.com/watch?v=YCzL96nL7j0) , utoljára megtekintve: 2024.11.19.

[12] Kumar, R. Evaluation metrics to check performance of regression models: (https://www.enjoyalgorithms.com/blog/evaluation-metrics-regression-models), utoljára megtekintve: 2024.11.19.

[13] Scikit learn: SelectKBest dokumentációja: (https://scikit-learn.org/1.5/modules/generated/sklearn.feature_selection.SelectKBest.html), utoljára megtekintve: 2024.11.19.

[14] Scikit learn: f_regression dokumentációja: (https://scikit-learn.org/1.5/modules/generated/sklearn.feature_selection.f_regression.html#sklearn.feature_selection.f_regression), utoljára megtekintve: 2024.11.19.

[15] Macrotrends: 2008-as populációs adat beszerzéséhez: (https://www.macrotrends.net/global-metrics/countries/USA/united-states/population), utoljára megtekintve: 2024.11.19.

Köszönetnyilvánítás:

Szeretném kifejezni őszinte hálámat témavezető tanáromnak Dr. Harangi Balázs egyetemi docensnek, hogy tanácsaival segítette a szakdolgozatom elkészülését. Szakértelme és türelme nélkül ez a szakdolgozat nem születhetett volna meg.

Hálásan köszönöm Borza Szabolcsnak, angol-magyar tanár szakos hallgatónak, hogy alaposan átnézte a dolgozatomat, és segített a formai megjelenés tökéletesítésében, valamint a nyelvi helyesség biztosításában.

Szeretném kifejezni hálámat édesanyámnak, aki a dolgozat készítése során végig támogató jelenlétével segített, és erőt adott a nehéz pillanatokban.