# Csoporttagok:

* Petrik Ádám (GVERV7)
* Balog Barnabás (YRVN14)
* Szilveszter Milán (CZZYLG)
* Gergely Jácint (NPCAPG)

# Regressziós feladat:

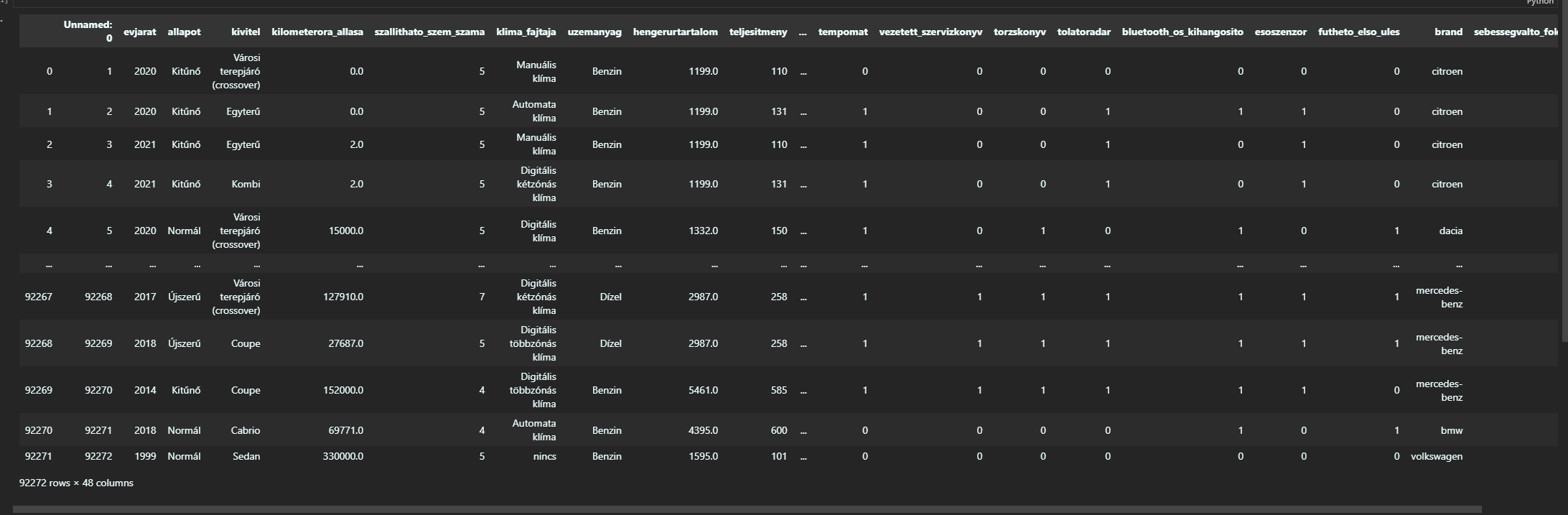
### „Nagy” adathalmaz:

Az adathalmaz a használtautó.hu autókereskedő oldalra feltöltött hirdetések adataiból áll. Az adathalmaz kb. 92 ezer rekordot és 48 attribútumot tartalmaz. A modell célja, hogy az elérhető paraméterekből megbecsüljük az egyes gépjárművek vételárát.

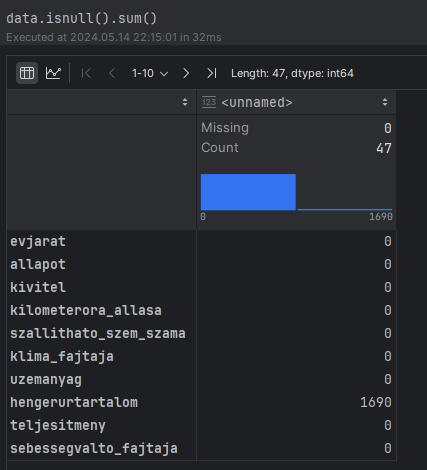
Adattisztítási lehetőségek:

* Karakterkódolási hibák javítása
* Üres értékek helyettesítése
* Szöveges értékek számokká alakítása

**Adatforrás:** <https://www.kaggle.com/datasets/nndorszakmry/hasznltautk-adatai?resource=download>

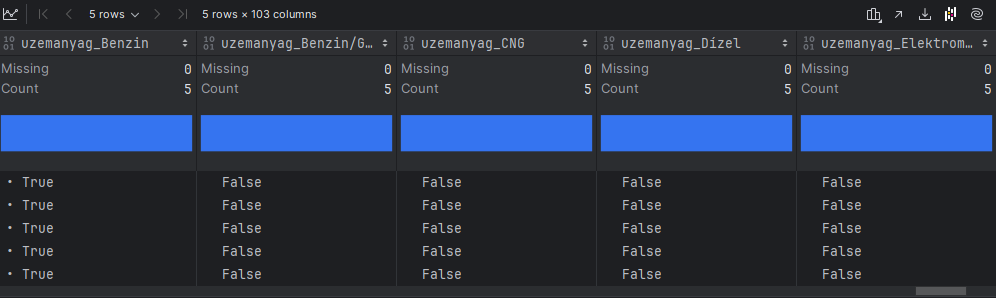
****

Miután beolvastuk az adatokat, ellenőriztük, hogy szükség van-e adattisztításra.

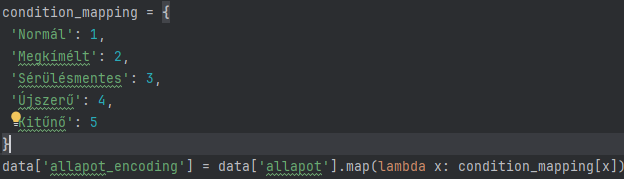


Mivel az elektromos autóknak nincs hengerűrtartalma, így ezeket 0 értékkel töltöttük fel.

A márka oszlopot és az üzemanyag oszlopot felosztottuk külön oszlopokra a **pd.get\_dummies** segítségével. Erre azért volt szükség, mert a regressziós modellek számértékeket várnak input változóként. Ezzel elértük azt, hogy például az üzemanyag típusokra létrejött egy-egy oszlop és amelyik kategóriába tartozik az adott autó ott True az érték. (Ezeket az értékeket már fel tudja ismerni a modell (0/1) és különbséget tud tenni a márkák és üzemanyag típusa között)



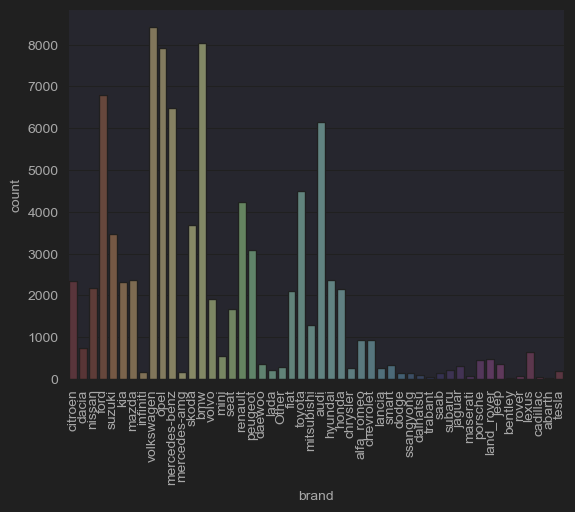
Az autók állapotának is szöveges reprezentációja van az adatok között. Ezeket az értékeket, mivel egy skálát tudunk belőlük alkotni számértékekké konvertáltuk:

****

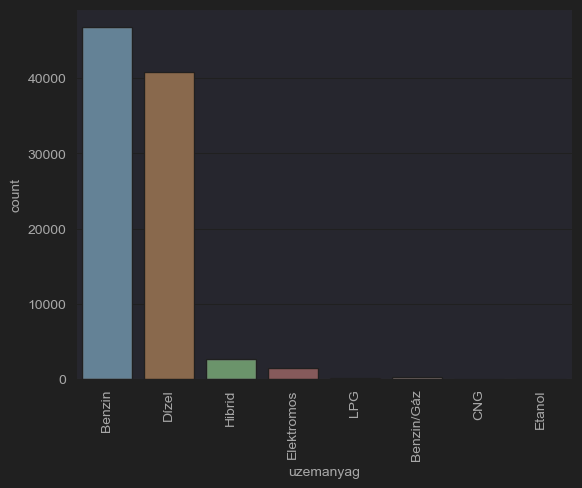
Így olyan számértékeket kaptunk, ahol a szám reprezentálja az állapotot. Minél nagyobb a szám értéke, annál jobb az autó állapota.

**Adatok vizualizációja**

A hirdetett autók darabszáma márkánként



A hirdetett autók darabszáma üzemanyag szerint



**Választott ML algoritmusok:**

* LinearRegression
* Lasso
* Ridge
* DecisionTreeRegressor
* RandomForestRegressor

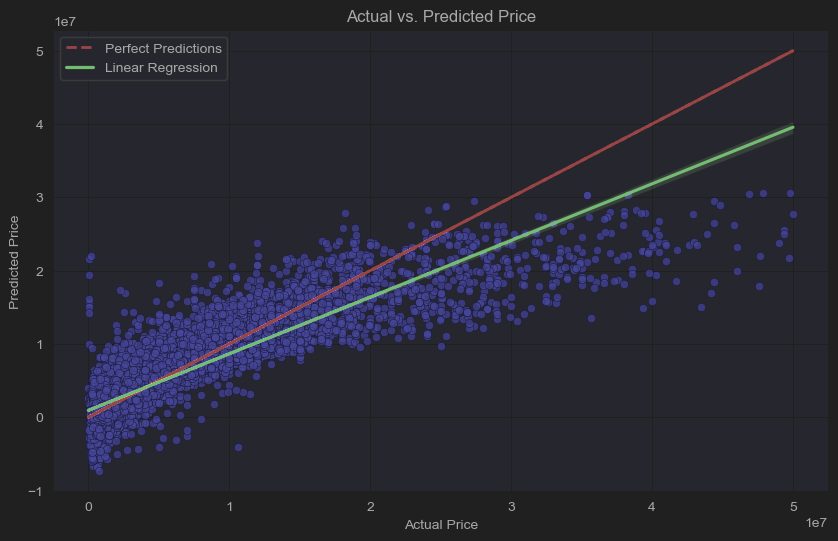
**Tanítás és kiértékelés:**

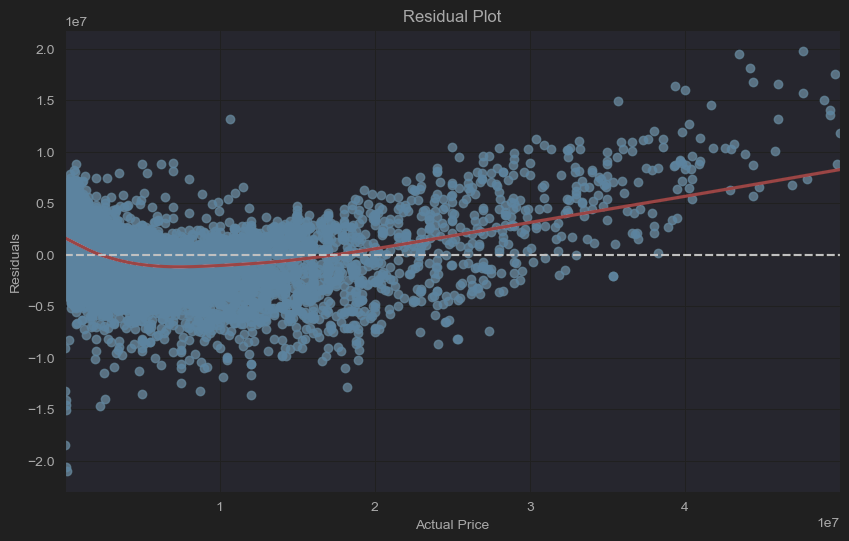
Az adathalmazt felosztottuk tanuló és teszt adatokra az Skicit Learn train\_test\_split() metódusával. Majd a különböző algoritmusokat használva a következő eredményekre jutottunk:

|  |  |
| --- | --- |
| **Eredmények** | |
| **Gépi tanuló algoritmus** | **Pontosság** |
| LinearRegression | 0.7671041674475926 |
| Ridge | 0.7671132515190189 |
| Lasso | 0.7671041606337548 |
| DecisionTreeRegressor | 0.8556583099441392 |
| RandomForestRegressor | 0.8752826786859957 |

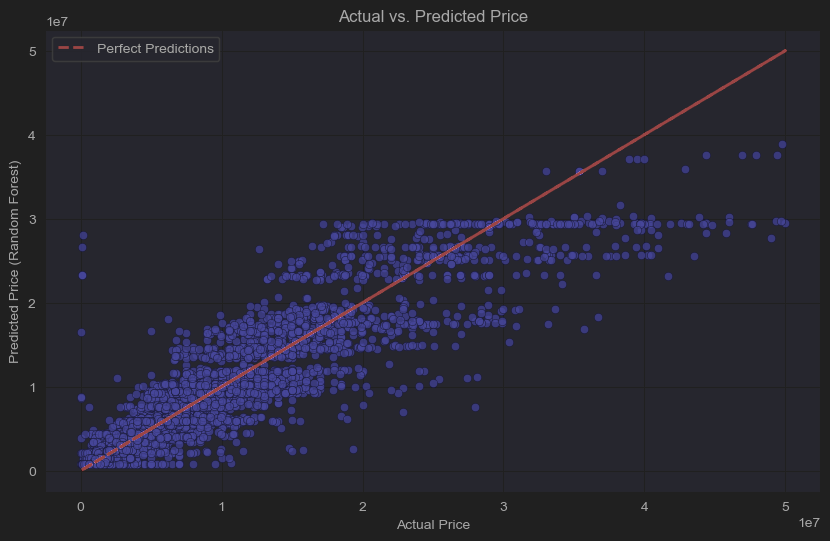
**Eredmények vizualizációja:**

LinearRegression:





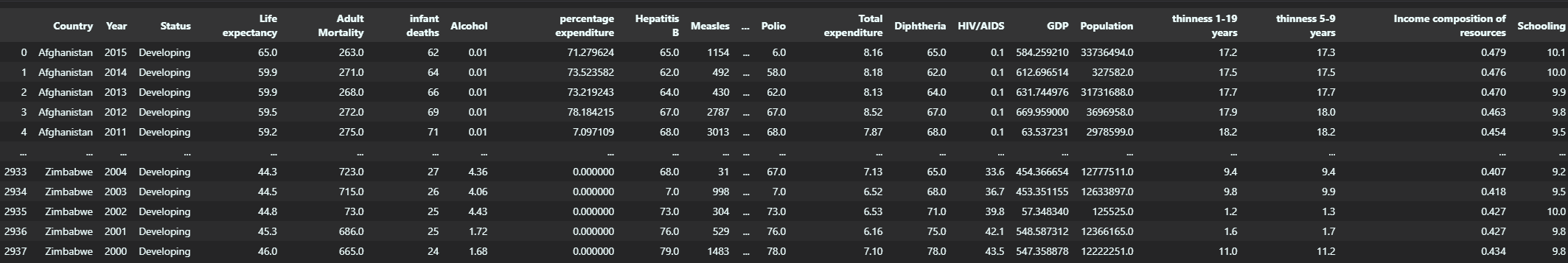
RandomForestRegressor:



### „Kis” adathalmaz:

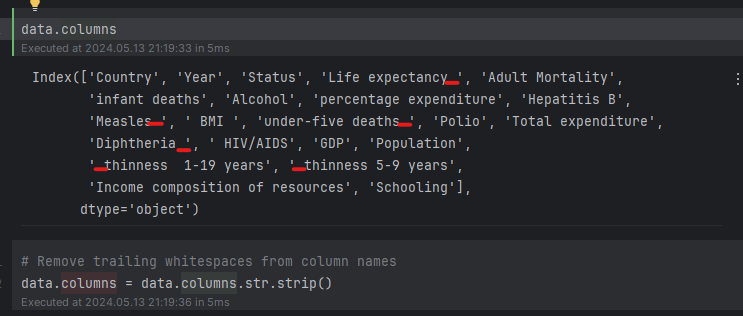
Annak az elemzésére szolgál az adathalmaz, hogy melyek azok a változók, amelyek kulcsfontosságú szerepet játszanak a várható élettartam alakításában. Az adathalmaz közel 3000 rekordot tartalmaz 22 oszloppal.

**Adatforrás**: <https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who/data>

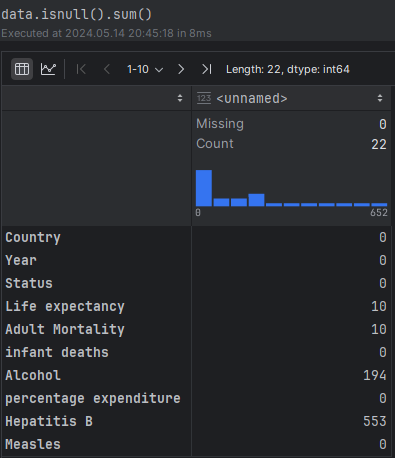


Miután beolvastuk az adatokat, megtisztítottuk őket, hibás adatokat javítottuk.

**Példa:**

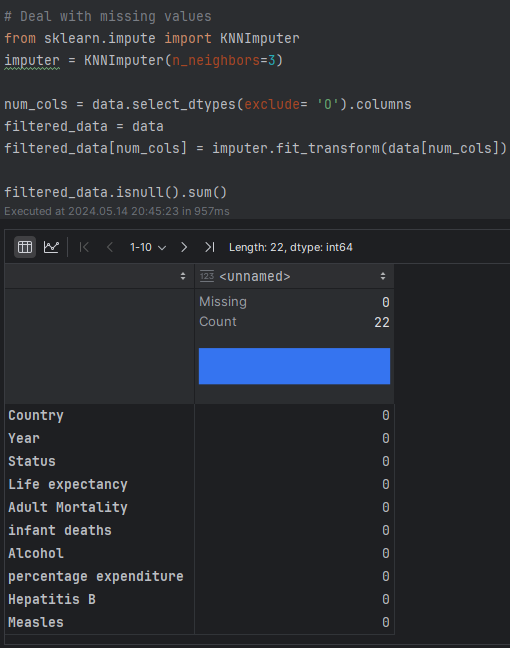


**Hiányzó értékek kezelése:**



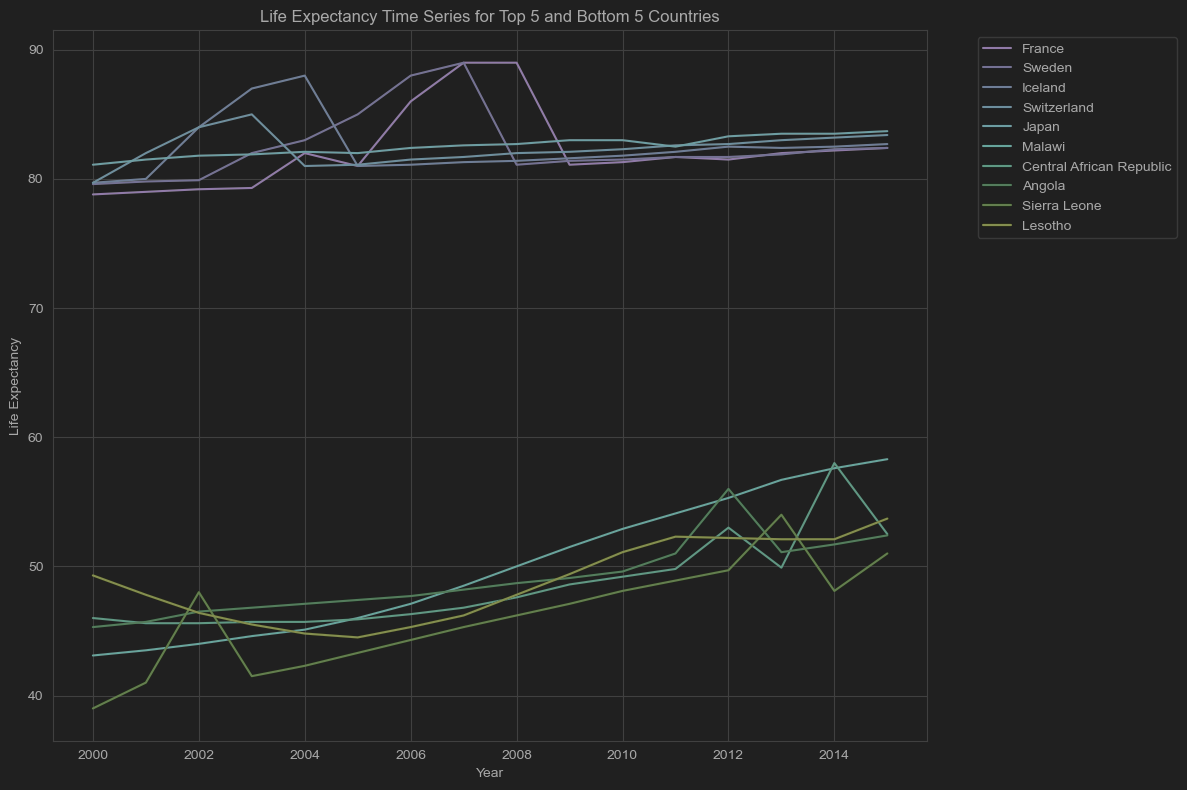
Ha a hiányzó értékek sorait egyszerűen csak törölnénk akkor az adatszettünk a felére csökkenne, így nem lenne reprezentatív az adatunk.

A hiányzó adatok kezelésére az **sklearn.impute** package **KNNImputer** modulját használtuk:

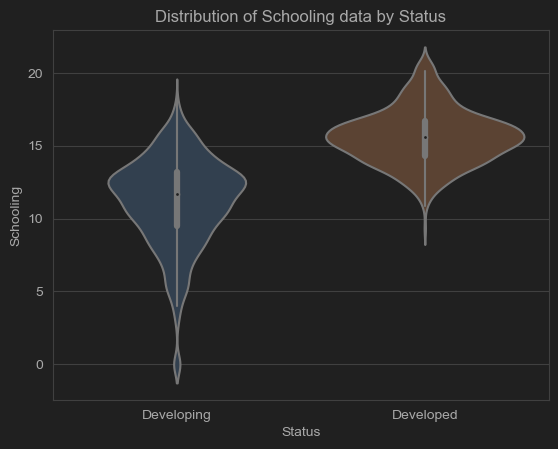


**Adatok vizualizációja:**

Várható élettartam szempontjából az 5 legjobban és legrosszabban teljesítő ország



Iskolázottság eloszlása fejlett és fejlődő országokban:



A diagramok szélessége arra utal, hogy a fejlett országokban jóval magasabb a sűrűsége azoknak, akik több mint 15 év iskolát végeznek, mint a fejlődő országokban.

**Választott ML algoritmusok:**

* LinearRegression
* Lasso
* Ridge
* DecisionTreeRegressor
* RandomForestRegressor

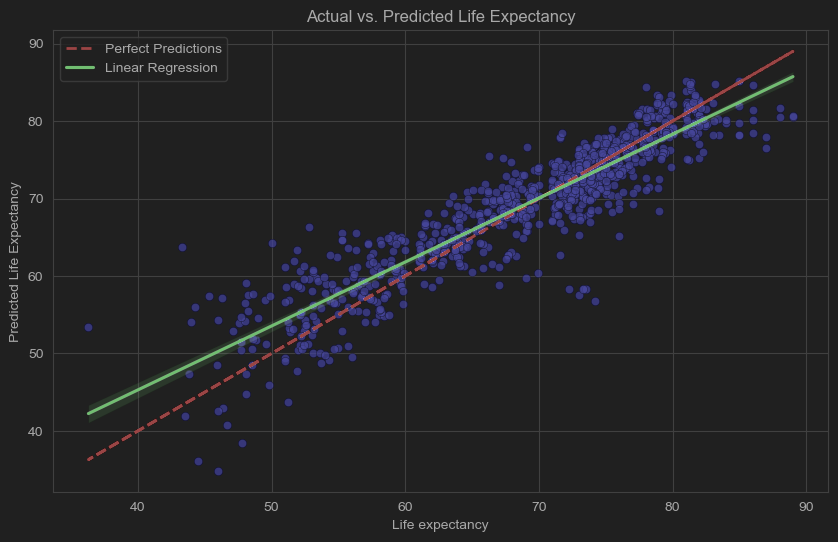
**Tanítás és kiértékelés:**

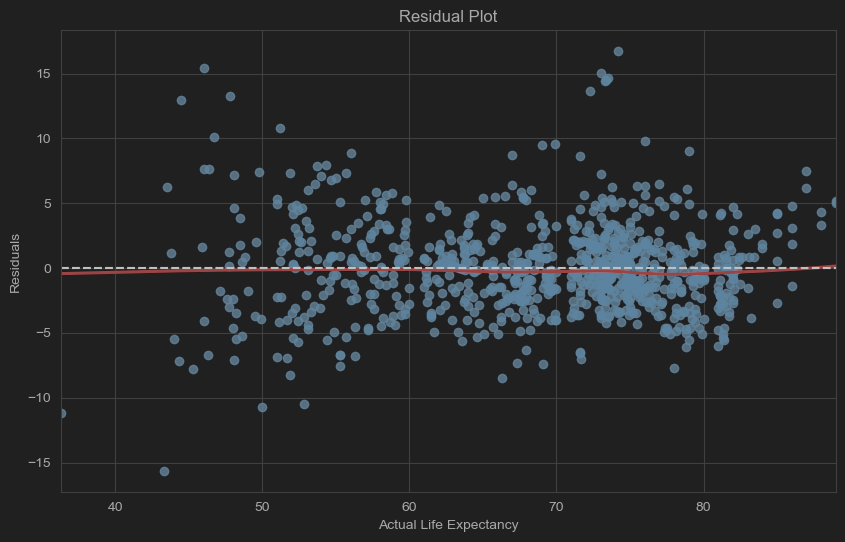
Az adathalmazt felosztottuk tanuló és teszt adatokra az Skicit Learn train\_test\_split() metódusával. Majd a különböző algoritmusokat használva a következő eredményekre jutottunk:

|  |  |
| --- | --- |
| **Eredmények** | |
| **Gépi tanuló algoritmus** | **Pontosság** |
| LinearRegression | 0.8357861997510714 |
| Ridge | 0.8358051000330954 |
| Lasso | 0.8280164129060429 |
| DecisionTreeRegressor | 0.8821965076926579 |
| RandomForestRegressor | 0.9286977571090572 |

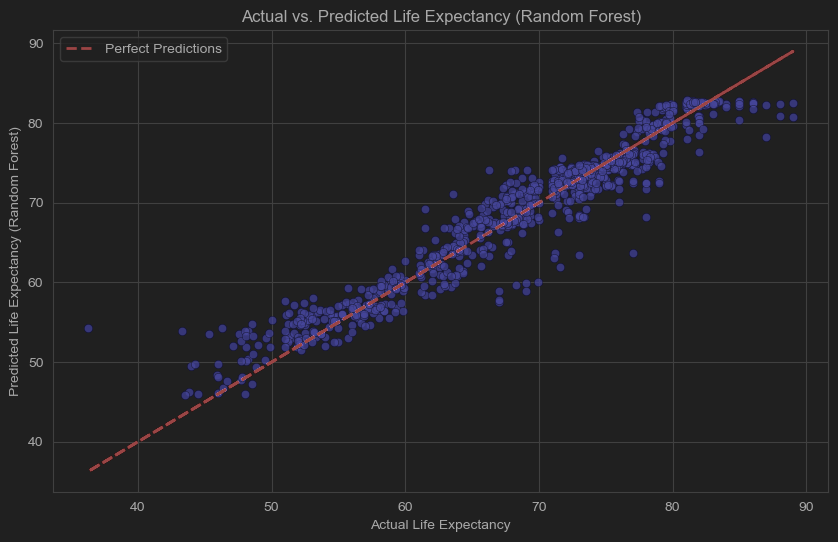
**Eredmények vizualizációja:**

LinearRegression:

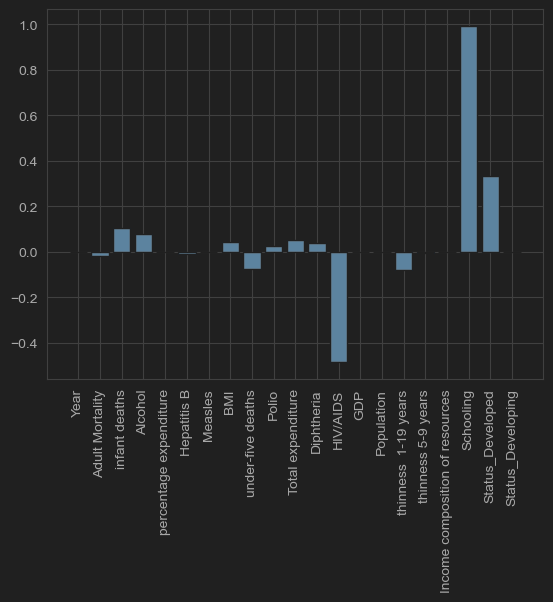




RandomForestRegressor:



Megvizsgáltuk azt is, hogy melyek a főbb befolyásoló feature-ök. az adatszettben. Ezek a tulajdonságok értékei befolyásolják a legjobban a várható élettartamot.

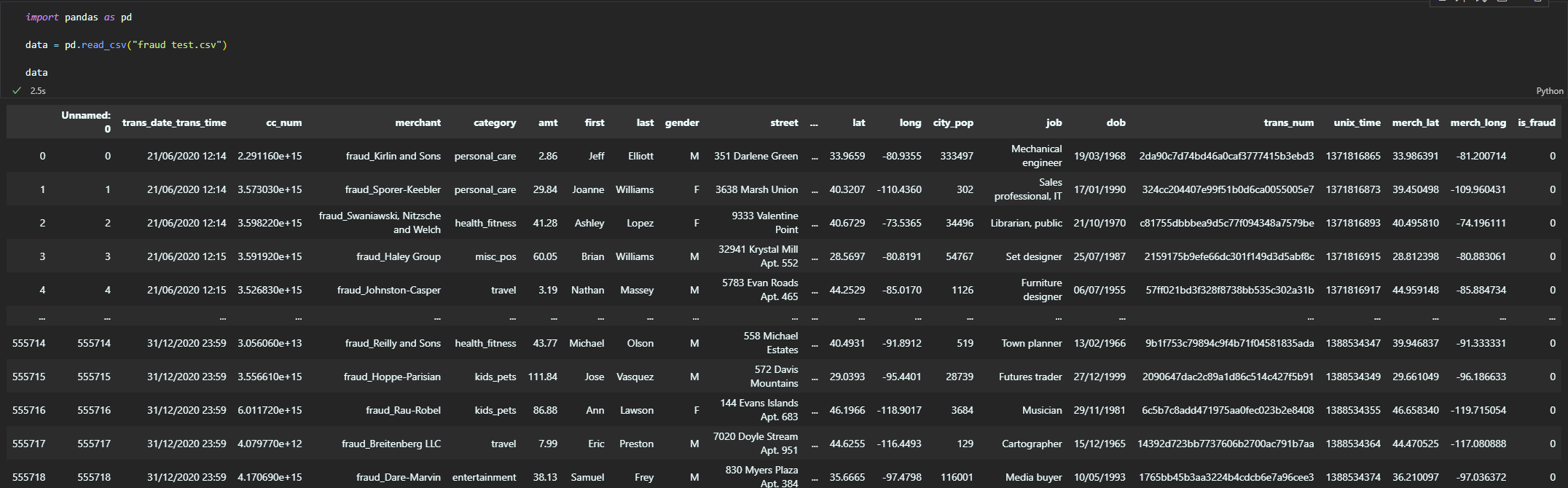


# Osztályozási feladat:

### „Nagy” adathalmaz:

A modell célja a hamis hitelkártya-tranzakciók felderítése, annak érdekében, hogy az ügyfeleket ne terheljék olyan tételekkel, amelyeket nem ők vásároltak. Az adathalmaz több mint 550 ezer rekordot és 23 attribútumot tartalmaz.

**Adathalmaz:** <https://www.kaggle.com/datasets/kelvinkelue/credit-card-fraud-prediction/data>



**Vizsgált ML algoritmusok:**

* Logistic Regression,
* SVM,
* Decision Tree,
* Random Forest,
* Naive Bayes

**Adatok vizsgálata**

Az adatok vizsgálata során több információt is sikerült kideríteni:

* Az adathalmaz üres/null értékeket nem tartalmazott,
* több olyan oszlop is előkerült, amely nem befolyásolja a célváltozót (pl. név, tranzakció ideje, tranzakció id stb.),
* lényegesen több valid tranzakció van, mint fraud, amely a modell predikcióit befolyásolhatja.

**Adatok előkészítése a tanításhoz**

Az előzőleg említett vizsgálat alapján az adatelőkészítés az alábbi lépésekből áll:

* Nem használható, irreleváns oszlopok eldobása.
* A lényegesen több valid adatot sampling segítségével redukáltam annyira, hogy egyezzen a fraud tranzakciók számával. Az adathalmaz a sampling után 2145 valid és fraud tranzakciót tartalmaz (összesen 4290 rekord).
* Feature Engineering: Több egymáshoz kapcsolódó oszlopot egybevontam, a dimenzionlitás csökkentése és a modell teljesítményének növelése érdekében (pl. helyadatok).

**Tanítás és kiértékelés**

Az adathalmazt tanítás előtt szétbontottuk tanító és tesztelő adathalmazokra a *Scikit Learn train\_test\_split()* metódusának segítségével, majd egy *LabelEncoder* segítségével a kategorikus szöveges oszlopokat numerikus oszlopokká alakítottam. Ezután *StandardScaler*-rel standardizáltam a rekordok szórását.

Az algoritmusokat a vizsgálat során először a teljes adathalmazon futtattam az alapvető paraméterekkel, majd a redukált adathalmazon, ahol már *GridSearchCV* segítségével kerestem meg egy előre meghatározott paraméter dict-ből a legjobb eredményt hozó paramétereket.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Pontosság (default params)** | | **Pontosság (GridSearchCV)** |
| **ML algoritmus** | **Teljes adathalmaz** | **Redukált adathalmaz** | **Redukált adathalmaz** |
| Logistic Regression | 87% | 100% | 85,1% |
| SVM | - | 88% | 85,2% |
| Naive Bayes | 99% | 88% | 84,82% |
| Decision Tree | 100% | 94% | 94,49% |
| Random Forest | 100% | 96% | 95,45% |

Mint, ahogy az adatok vizsgálata során említettem az eredményekből is látszik, hogy a teljes adathalmazon feltanított algoritmusok az esetek többségében túlillesztettek (sárga kijelölés), gyanúsan jó eredményeket hoztak. Az esetek többségébe a legjobb eredményeket a redukált adathalmazokkal értem el, ahol biztosan tudtam, hogy nem történt túlillesztés. Egyedül a döntési fának tudtam olyan paraméterkombinációt találni, amely jobb eredményt ért el, mint az alapbeállítás.

A GridSearchCV által talált legjobb paraméterkombinációk a mellékelt notebookban találhatóak.

### „Kis Adathalmaz”:

Ez az adatkészlet az Agaricus és Lepiota családba tartozó 23 kopoltyús gombafajnak megfelelő hipotetikus minták leírását tartalmazza. Minden egyes faj azonosítva van, mint biztosan ehető, biztosan mérgező, vagy ismeretlen ehetőségű és nem ajánlott.

A feltanított modell célja a különböző gombákról eldönteni, hogy mérgezők vagy ehetőek-e.

Az adathalmaz 8123 rekordot és 23 oszlopot tartalmaz.

A képen képernyőkép látható

Automatikusan generált leírás

**Adatforrás**: <https://archive.ics.uci.edu/dataset/73/mushroom>

**Vizsgált ML algoritmusok:**

* Logistic Regression,
* KNN,
* SVM,
* Decision Tree,
* Random Forest,
* Naive Bayes

**Adatok vizsgálata**

A vizsgálat során kiderült, hogy:

* az adathalmaz oszlopai csak szöveges értékeket tartalmaznak,
* a ’stalk-root’ oszlop 2480 üres értéket tartalmaz,
* az adatok tisztított, összeszedett formában vannak jelen.

Mivel az adathalmazzal különös problémát (sok hiányzó érték, kódolási hiba stb.) nem figyeltem meg, így az adattisztítással nem kell foglalkoznom, hiszen a talált hiányzó értékek nem adatbeviteli hibából adódóan hiányoznak, hanem helyes értékek.

**Adatok előkészítése a tanításhoz**

Az adatok előkészítése csak egy lépést foglal magában: az oszlopok értékeit normalizálom úgy, hogy a szöveges értékek helyett szám értékeket rendelek a *Scikit Learn* könyvtár *LabelEncoder* osztályának használatával. *Feature Scaling*-et nem szükséges alkalmazni, mert az oszlopok így is eléggé kis varianciával rendelkeznek.

**Tanítás és kiértékelés**

Az adathalmazt tanítás előtt szétbontottuk tanító és tesztelő adathalmazokra a *Scikit Learn train\_test\_split()* metódusának segítségével, majd először minden algoritmust az alapvető beállításaival „tanítottam” fel.

|  |  |
| --- | --- |
| **Eredmények** | |
| **Gépi tanuló algoritmus** | **Pontosság** |
| Logistic Regression | 95,57% |
| KNN (k=1) | 99,95% |
| SVM | 100% |
| Naive Bayes | 93,11% |
| Decision Tree | 100% |
| Random Forest | 100% |

A legjobban teljesítő algoritmusok az *SVM, Decision Tree* és *Random Forest* voltak. Részletesebben ezek közül a Decision Tree algoritmust vizsgáltam meg, hiszen minél mélyebb a fa struktúrája, annál inkább hajlamos az „overfittingre”.   
Ehhez egy ciklus segítségével végig néztem, hogy hogyan változik a modell pontossága egyre növekvő mélység mellett:

A képen szöveg, Betűtípus, képernyőkép látható

Automatikusan generált leírás

A legjobb mélység, ahol még biztos nincs túlillesztve az algoritmus a 7.