Načítať dáta

Najprv načítam údaje rovnakou jednoduchou metódou ako v predchádzajúcej aplikácii.

Potom som odstránil odľahlé hodnoty. Považoval som za dôležité odstrániť jednotlivé odľahlé hodnoty v nasledujúcich stĺpcoch: Price, Prod. Year, Engine volume, Cylinders, Airbags.

Ako vidíte na obrázku, stalo sa to nasledovným spôsobom. Existovali odľahlé hodnoty, ktoré sa zdali realistické, takže hodnoty jednotlivých stĺpcov boli regulované iným spôsobom.

```
df = df[df['Price'] >= 950]
df = df[df['Price'] <= 500000]
df = df[df['Prod. year'] >= 1950]
df = df[df['Engine volume'] >= 0]
df = df[df['Cylinders'] >= 0]
df = df[df['Airbags'] >= 0]
```

Obrázok 1. Outliery

```
******************** Before removing outliers ********************
---------- Min ----------
ID               20746880
Price                1.0
Prod. year        1939
Engine volume        0.0
Cylinders              1
Airbags                0
Turbo engine       False
Left wheel         False
dtype: object
---------- Max ----------
ID                45816654
Price            26307500.0
Prod. year             2020
Engine volume          20.0
Cylinders                16
Airbags                  16
Turbo engine           True
Left wheel             True
dtype: object
```

Obrázok 2. Min., max pred odstránením

```
******************** After removing outliers ********************
---------- Min ----------
ID               24367759
Price               950.0
Prod. year           1953
Engine volume         0.0
Cylinders               1
Airbags                 0
Turbo engine        False
Left wheel          False
dtype: object
---------- Max ----------
ID               45816654
Price             308906.0
Prod. year            2020
Engine volume         20.0
Cylinders               16
Airbags                 16
Turbo engine          True
Left wheel            True
dtype: object
```

Obrázok 3. Min., max po odstránením

Potom som skonvertoval hodnoty Levy na typ float, pretože sa mi už stalo predvídateľným, že v tomto stĺpci bude veľa hodnôt chýbať. Keď sa to stalo, skontroloval som, či je v tomto stĺpci veľa nulových hodnôt. A áno, bolo toho veľa. Už som vedel, že budem musieť odstrániť stĺpec Levy.

Na obrázku môžete vidieť, koľko nulových hodnôt je v stĺpci Levy.

```
******************** Missing values *****
Lenght of dataset: 16660
ID                     0
Price                  0
Levy                5439
Manufacturer           0
Model                  0
Prod. year             0
Category               0
```

Obrázok 4. Null hodnoty

Potom som sa pozrel na dátové typy všetkých stĺpcov, aby som zistil, či musím ešte niečo zmeniť. A áno, musela som. Zmenila som typ údajov o Mileage. Urobil som to typu int takto:

```
print(f"Mileage data type: {df['Mileage'].dtype}")
df['Mileage'] = df['Mileage'].str.replace(' km', '').astype(int)
print(f"Mileage data type: {df['Mileage'].dtype}")
df = df[df['Manufacturer'] != 'ЬЂЗ']
```

Obrázok 5. Kód pre Mileage

```
Mileage data type: object
Mileage data type: int32
```

Obrázok 6. Dátový typ Mileage

Zo súboru údajov som tiež odstránil ID a duplikáty. Prvý obrázok ukazuje, aké stĺpce tam boli pred odstránením ID a pod ním môžete vidieť, kedy bol stĺpec ID odstránený.

```
******************** Before removing ID ********************
Index(['ID', 'Price', 'Levy', 'Manufacturer', 'Model', 'Prod. year',
       'Category', 'Leather interior', 'Fuel type', 'Engine volume', 'Mileage',
       'Cylinders', 'Gear box type', 'Drive wheels', 'Doors', 'Color',
       'Airbags', 'Turbo engine', 'Left wheel'],
      dtype='object')
******************** After removing ID ********************
Index(['Price', 'Levy', 'Manufacturer', 'Model', 'Prod. year', 'Category',
       'Leather interior', 'Fuel type', 'Engine volume', 'Mileage',
       'Cylinders', 'Gear box type', 'Drive wheels', 'Doors', 'Color',
       'Airbags', 'Turbo engine', 'Left wheel'],
      dtype='object')
```

Obrázok 7. Odstránenie ID

Na ďalšom obrázku uvidíte, koľko údajov bolo s duplikátmi a po odstránení duplikátov koľko údajov zostalo.

```
******************** Before drop duplicate ********************
Lenght of dataset: 16660
******************** After drop dupicate ********************
Lenght of dataset: 14327
```

Obrázok 8. Duplikáty

Odstránil som zostávajúcu položku zo súboru údajov s nasledujúcim riadkom kódu:

df = df[df['Manufacturer'] != ЬЂЗ]

Rozdelenie dáta na testovaciu a trénovaciu množinu

V ďalšom kroku som musela zvoliť vhodné kódovanie a preto som zvolila dummy kódovanie. Musela som sa rozhodnúť, ktoré stĺpce chcem kódovať. Musela som vziať do úvahy, že v každom stĺpci môže byť veľa rôznych hodnôt, takže som ich nemohol vybrať všetky a musela som niektoré stĺpce odstrániť, aby som mala správny počet hodnôt.

Preto som pre dummy vybrala nasledujúce stĺpce:

- Manufacturer
- Category
- Leather interior
- Fuel type
- Gear box type

Vybrala som jednotlivé stĺpce, na základe ktorých hodnoty najlepšie pokrývajú hlavný prvok, cenu.

Takže, aby sme mohli vykonať normalizáciu/zmenšenie, vieme, že musíme odstrániť ostatné prvky, ktoré nie sú založené na číslach. Preto som vymazala nasledujúce stĺpce:

- Model (Vymazal som to, pretože v tomto stĺpci je veľa rôznych hodnôt.)
- Levy (null hodnoty)
- Drive wheels
- Doors
- Color
- Turbo engine
- Left wheel

Malá časť našej množiny údajov po dummy kódovaní.

Hydrogen	LPG	Petrol	Plug-in Hybrid	Automatic	Manual	Tiptronic	Variator
0	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0
0	0	1	0	0	0	0	1
0	0	0	0	1	0	0	0
0	0	1	0	1	0	0	0
...	...	...	...	...	...	...	...
0	0	0	0	0	1	0	0
0	0	1	0	0	0	1	0
0	0	0	0	0	1	0	0
0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0

Obrázok 9. Dummy

Rozdelenie dáta na testovaciu a trénovaciu množinu

```
X = df.drop(columns=['Price'])
y = df['Price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1,
random_state=42
```

Tieto hodnoty výrazne ovplyvnia jednotlivé modely, takže ich možno neskôr zmeniť pre lepšie výsledky. V prvom kole nastavme veľkosť testu na 0,1 a náhodný stav na 42, pretože to je podľa mňa ideálne rozdelenie.

Po dokončení distribúcie som ju normalizovala v nasledujúcich riadkoch:

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Nasledujúci obrázok ukazuje, ako sú údaje rozdelené na testovanie a trénovanie:

```
X_train shape: (12892, 91)
X_test shape: (1433, 91)
y_train shape: (12892,)
y_test shape: (1433,)
```

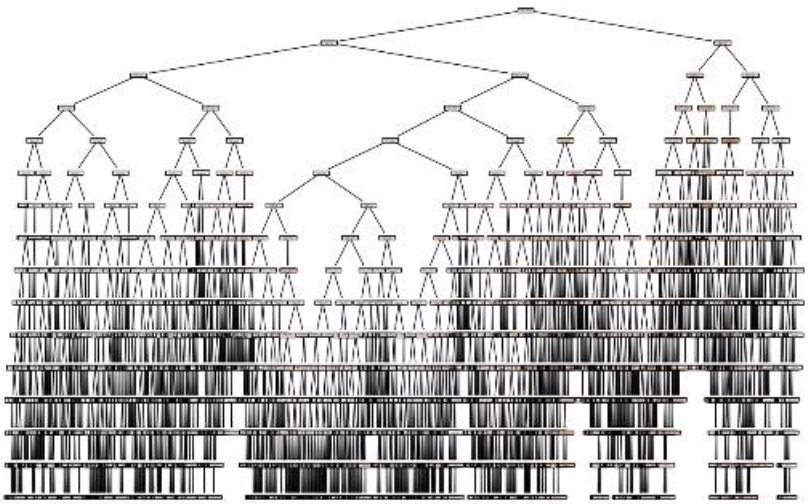
Obrázok 10. Rozdelenie na testovaciu a trénovaciu množinu

Rozhodovací strom

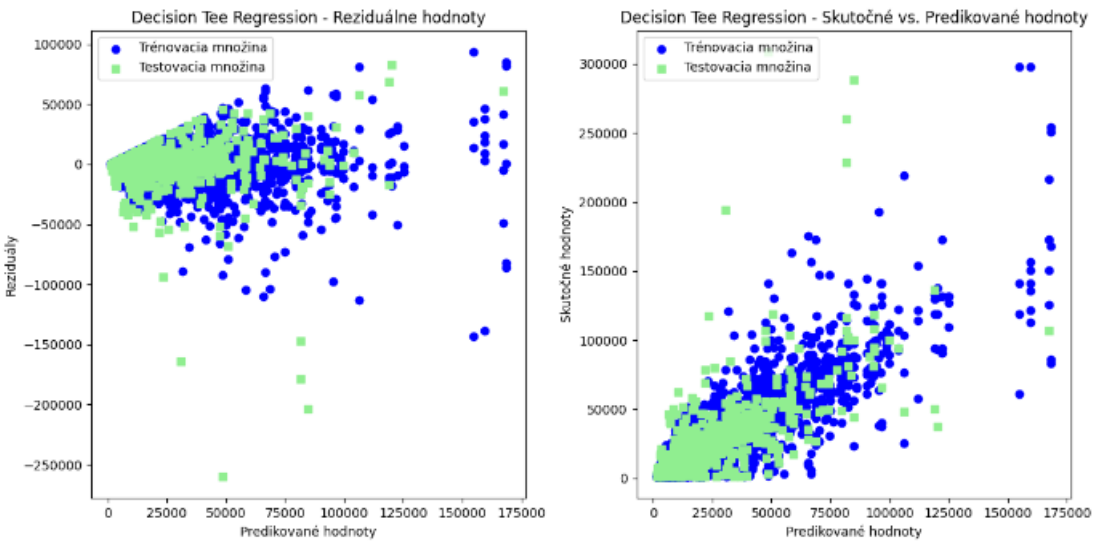
Pomocou nasledujúceho modelu a vhodnej distribúcie sa získal nasledujúci výsledok:

```
Decision tree
R2 Score: 0.5120121221058012
************************************************
Decision Tee Regression - Train
Mean Squared Error: 63804659.779644914
Root Mean Squared Error: 7987.781906114169
R2 Score: 0.806246873208491
Decision Tee Regression - Test
Mean Squared Error: 249491990.78823322
Root Mean Squared Error: 15795.315469728144
R2 Score: 0.5120121221058012
```

Obrázok 11. R2, MSE, RMSE decision tree



Obrázok 12. Strom decision tree



Obrázok 13. Reziduálne hodnoty decision tree

Dôvod, prečo je DecisionTree takmer nečitateľný, je ten, že som pracovala s nasledujúcimi údajmi, aby som dosiahla čo najlepšie skóre R2.

```
decision_tree_model = DecisionTreeRegressor(max_depth=15, min_samples_split=4,
min_samples_leaf=4, random_state=42)
decision_tree_model.fit(X_train, y_train)
y_pred = decision_tree_model.predict(X_test)
r2 = r2_score(y_test, y_pred)
```

Čo presne znamenajú získané hodnoty pre náš model DecisionTreeRegressor?

- Hodnota R2 ukazuje mieru prispôsobenia modelu. Keď máme hodnotu 1, to by bolo najideálnejšia.
- Mean Squared Error -MSE ukazuje kvalitu prispôsobenia meraním štvorcových odchýlok medzi hodnotami odhadnutými modelom a skutočnými hodnotami. Čím nižšia je hodnota MSE, tým lepšie je model prispôsobený údajom. Menší MSE znamená lepší výsledok.
- Root Mean Squared Error -RMSE je odvodením od druhej odmocniny MSE a nesie rovnaké informácie, ale zobrazuje výsledky v pôvodných jednotkách výstupnej premennej. Nižšie RMSE tiež znamená lepší výkon modelu.
- Konkúzia- Tu prezentované výsledky ukazujú, že regresný model rozhodovacieho stromu funguje celkom dobre na trénovacích údajoch (hodnota 0,806246873208491 R2), ale menej efektívne na testovacích údajoch (hodnota 0,5120121221058012 R2). Hodnoty RMSE sú vyššie na testovacích údajoch, čo znamená, že model poskytuje menej presné odhady testovacích údajov.

Ľavý graf: "Decision Tee Regression - Reziduálne hodnoty"

- Tento graf zobrazuje rozdiely (zvyšky) medzi hodnotami predpovedanými modelom a skutočnými hodnotami. Zvyšky sa vypočítajú odpočítaním predpokladanej hodnoty od skutočnej hodnoty pre každý dátový bod.
- Predikované vardy/X- Hodnoty predpovedané modelom.
- Zostatkové/Y- Rozdiely medzi skutočnými a predpokladanými hodnotami.
- Modré bodky- Tréningové údaje množiny údajov.

- Zelené štvorce- Testovacie údaje súboru údajov.

V ideálnom prípade by boli rezíduá náhodne rozdelené okolo hodnoty 0, čo by znamenalo, že model robí konzistentne dobré predpovede. V tomto prípade sú body umiestnené v dosť širokom rozsahu na osi y, čo naznačuje, že medzi predikciami modelu a skutočnými hodnotami existujú značné rozdiely.

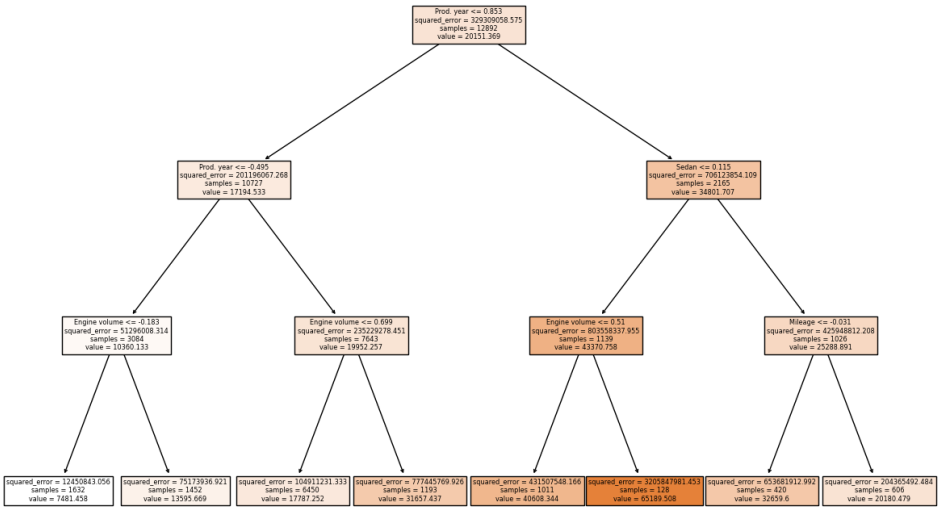
Pravý graf: „Regresia rozhodnutia Tee – skutočné vs. predpokladané hodnoty“

- Tento graf ukazuje porovnanie predpokladaných hodnôt so skutočnými hodnotami.
- Predikované vardy/X- Hodnoty predpovedané modelom.
- Skutočné vardy/Y- Skutočné hodnoty.
- Modré bodky- Tréningové údaje množiny údajov.
- Zelené štvorce- Testovacie údaje súboru údajov.

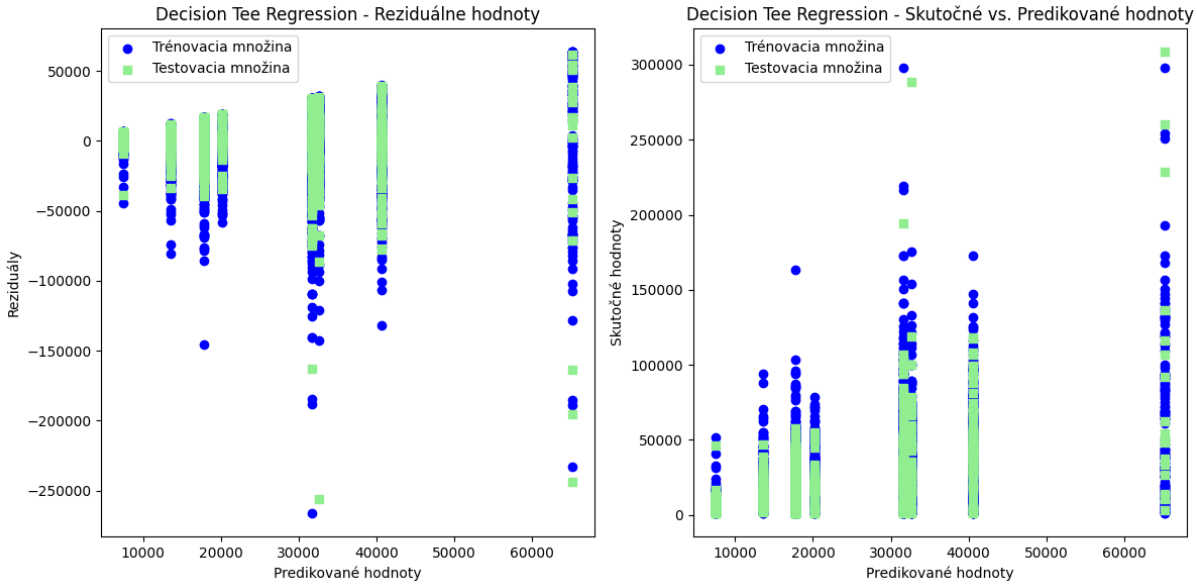
V ideálnom prípade by body ležali pozdĺž hlavnej uhlopriečky od ľavého dolného rohu k pravému hornému rohu, čo by znamenalo, že predpokladané a skutočné hodnoty sú rovnaké.

Teraz sa pozrime na príklad, ako čítať DecisionTreeRegressor, aby boli hodnoty jasne viditeľné na obrázku.

```
decision_tree_model = DecisionTreeRegressor(max_depth=3, min_samples_split=4,
min_samples_leaf=4, random_state=42)
```

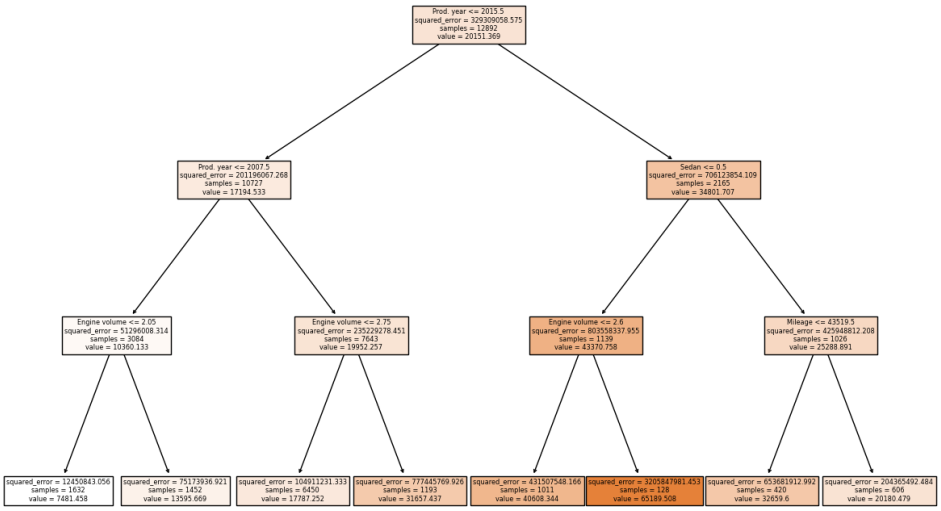


Obrázok 14. Rozhodovací strom



Obrázok 15. Reziduálne hodnoty decision tree

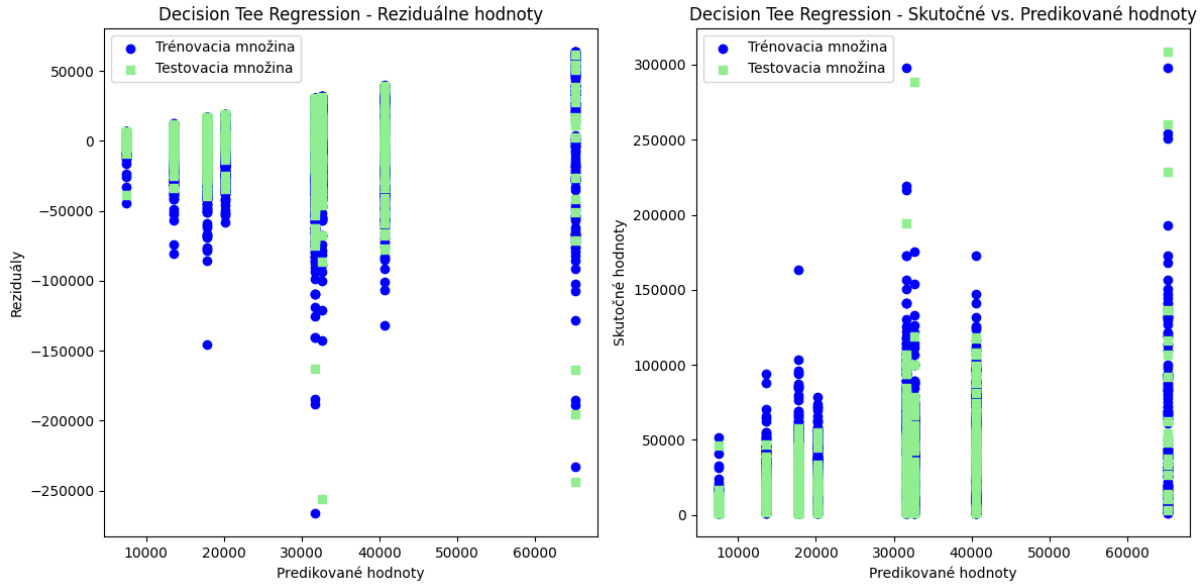
Tento strom nie je zrozumiteľný pretože je po normalizácií a nevidíme reálne roky, len desatinné čísla. Kvôli tomu skúšala som takto prerobiť kód aby nasledovaný strom bolo ešte pred normalizáciou, aby som vedela vyčítať jednotlivé podmienky. R2 skór bude horšie ako pri prvom grafy, lebo teraz max_depth je 3. Mean Squared Error/ Train: 231045388.0407378, Root Mean Squared Error/Train: 15200.177237148842, R2 Score/Train: 0.2983934634515246, Mean Squared Error/Test: 356011711.973327, Root Mean Squared Error/Test: 18868.272628233008, R2 Score/Test: 0.3036674272289379



Obrázok 16. Decision tree

Tu sa nám obrázok stáva oveľa čitateľnejším, pretože je pred normalizáciou, pracuje s tisíckou ešte nenormalizovaných údajov, čo je zrozumiteľnejšie, ako keď už pracujete s normalizovanými údajmi.

- squared_error-Táto hodnota ukazuje, o koľko sa vzorky pod daným vrcholom líšia od priemernej hodnoty. Čím je táto hodnota menšia, tým presnejšie model zodpovedá údajom v danom bode.
- Samples- udáva, koľko vzoriek patrí do skupiny definovanej vrcholom.
- Hodnota- táto hodnota zobrazuje priemernú hodnotu cieľovej premennej pre danú skupinu.
- V hlavnom vrchole Prod. podmienka platí pre rok. Otázka znie: Prod. rok je menej ako 2015,5? Ak áno, pravá časť zobrazuje pravdivú časť. Ak je to pravda, zmení sa na ďalšiu podmienku, ktorou nie je nikto iný ako Prod. rok menší alebo rovný 2007,5? Ak áno, tak opäť vezme vetvu pod tou, ktorá už bude skúmať objem motora, takže ak je objem motora menší alebo rovný 2,05, pozrie sa opäť na pravú vetvu pod tou. V tomto bode je šiška hotová.
- Na konci stromu sú listy (vrcholy najnižšej úrovne), ktoré sa ďalej nerozkladajú, ale obsahujú konečné predikčné hodnoty (value hodnoty).
- Pozrime sa na veci z druhej strany. Ak prejdeme z hlavného vrcholu na ľavú vetvu, vieme, že podmienka nebola splnená a pokúša sa nájsť inú podmienku. Hodnota sedanu musí byť menšia alebo rovná 0,5. Ak to nie je pravda, začne skúmať stav najazdených kilometrov, ale ak je pravdivý, bude skúmať stav súvisiaci s objemom motora.
- V celej veci je aj taká súvislosť, že by sme sa mali pozerať na hodnotu vzoriek na hlavnom vrchole. vzorka=12892. Ak pripočítame vzorové hodnoty dvoch vetiev pod jednu, výsledok je presne 12892.



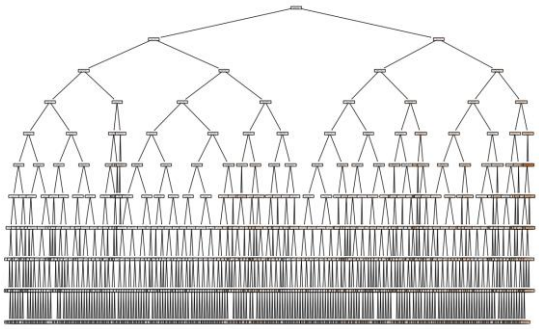
Obrázok 17. Reziduálne hodnoty decision tree

Random forest

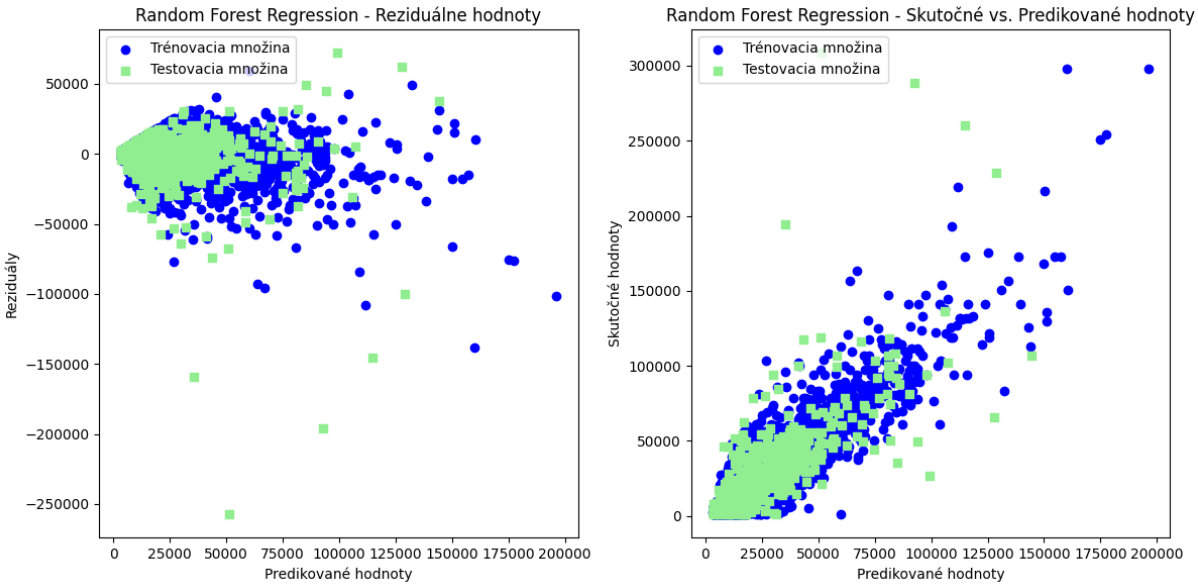
Pomocou nasledujúceho modelu a vhodnej distribúcie sa získal nasledujúci výsledok:

```
Bagging random forest
R2 Score: 0.5955569501431877
************************************************
Random Forest Regression - Train
Mean Squared Error: 59610725.80353525
Root Mean Squared Error: 7720.798262066899
R2 Score: 0.8189824292671666
Random Forest Regression - Test
Mean Squared Error: 206778295.60987204
Root Mean Squared Error: 14379.78774564743
R2 Score: 0.5955569501431877
```

Obrázok 18. Random forest R2, MSE, RMSE



Obrázok 19. Prvý strom z Random Forestu



Obrázok 20. Reziduálne hodnoty random forest

Kód na nastavenie:

```
random_forest_model = RandomForestRegressor(n_estimators=50, max_depth=10,
min_samples_split=2, min_samples_leaf=2, random_state=42)
random_forest_model.fit(X_train, y_train)
y_pred = random_forest_model.predict(X_test)
```

Je dôležité si uvedomiť, že tu sme mali celý les a strom zobrazený na obrázku je strom z lesa.

Ľavý graf: "Random Forest Regression - Reziduálne hodnoty"

- Predikované várdy/X- hodnoty predpovedané modelom.
- Zostatkové/Y- rozdiely medzi skutočnými a predpokladanými hodnotami.
- Modré bodky- tréningové údaje množiny údajov.
- Zelené štvorce- testovacie údaje súboru údajov.

V ideálnom prípade by boli rezíduá náhodne rozdelené okolo hodnoty 0, čo by znamenalo, že model robí konzistentne dobré predpovede. V tomto prípade sú body umiestnené v dosť širokom rozsahu na osi y, čo naznačuje, že medzi predikciami modelu a skutočnými hodnotami existujú značné rozdiely.

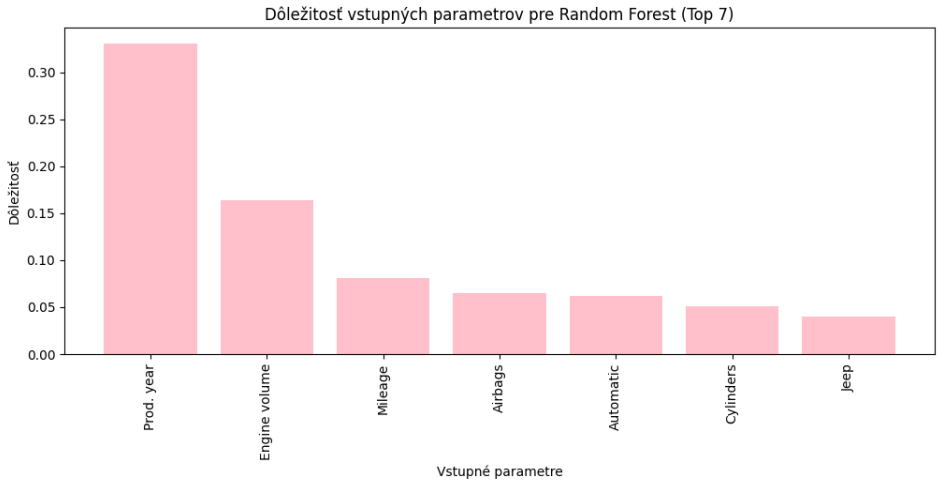
Pravý graf: „Regresia rozhodnutia Tee – skutočné vs. predpokladané hodnoty“

- Tento graf ukazuje porovnanie predpokladaných hodnôt so skutočnými hodnotami.
- Predikované várdy/X- hodnoty predpovedané modelom.
- Skutočné várdy/Y- skutočné hodnoty.
- Modré bodky-tréningové údaje množiny údajov.
- Zelené štvorce- testovacie údaje súboru údajov.

V ideálnom prípade by body ležali pozdĺž hlavnej uhlopriečky od ľavého dolného rohu k pravému hornému rohu, čo by znamenalo, že predpokladané a skutočné hodnoty sú rovnaké.

Získali sme podobný graf ako v predchádzajúcom príklade pre DecisionTreeRegressor. Minimálne rozdiely, ale oba grafy poskytli podobné výsledky.

Vizualizácia Random Forest



Obrázok 21. Vizualizácia Random Forest

Obrázok jasne ukazuje 7 najdôležitejších hodnôt pre Random Forest, ktorými nie sú nič iné ako:

- Prod. Year
- Engine volume
- Mileage

- Airbags
- Automatic
- Cylinders
- Jeep

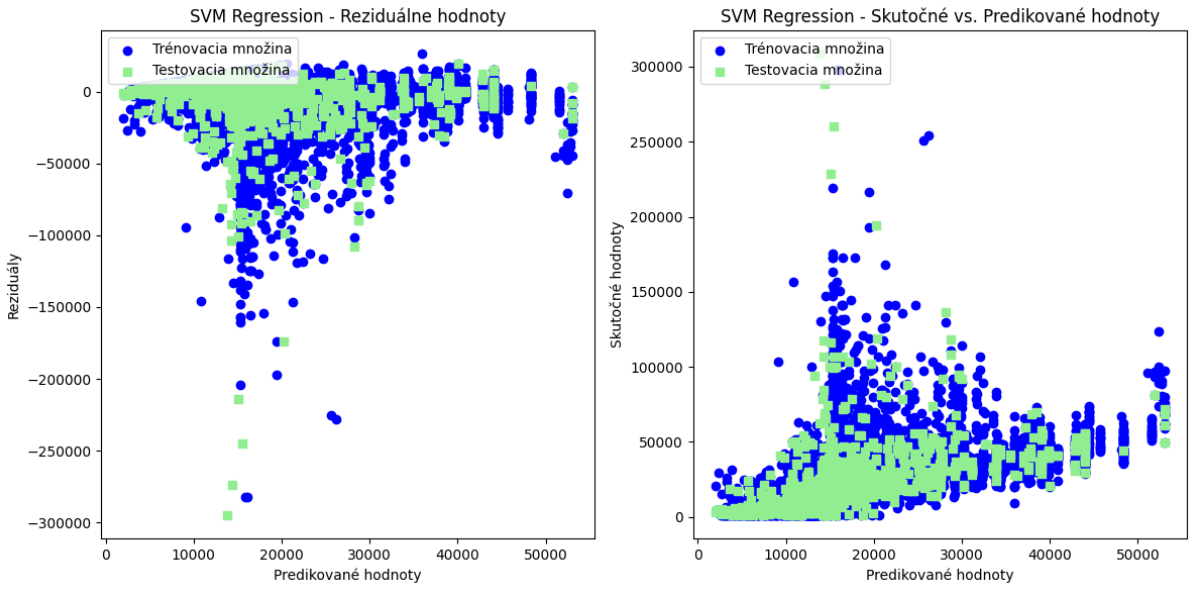
Príslušné hodnoty možno zhruba odčítať z osi y. Napríklad: Prod. year má najväčší vplyv, jeho dôležitosť meraná číslami je viac ako 0,3.

SVM

Pomocou nasledujúceho modelu a vhodnej distribúcie sa získal nasledujúci výsledok:

```
SVM
R2 Score: 0.13896113144695066
**************************************************
SVM Regression - Train
R2 Score: 0.27511450988792074
SVM Regression - Test
R2 Score: 0.13896113144695066
**************************************************
SVM Regression - Train
Mean Squared Error: 238711358.32337582
Root Mean Squared Error: 15450.286674472283
R2 Score: 0.27511450988792074
SVM Regression - Test
Mean Squared Error: 440220569.38866013
Root Mean Squared Error: 20981.43392117565
R2 Score: 0.13896113144695066
```

Obrázok 22. SVM R2, MSE, RMSE



Obrázok 23. Reziduálne hodnoty SVM

Ľavý graf: "SVM Regression - Reziduálne hodnoty"

- Predikované várdy/X- hodnoty predpovedané modelom.
- Zostatkové/Y-rozdiely medzi skutočnými a predpokladanými hodnotami.
- Modré bodky- tréningové údaje množiny údajov.
- Zelené štvorce- testovacie údaje súboru údajov.

Pravý graf: „SVM – skutočné vs. predpokladané hodnoty“

- Tento graf ukazuje porovnanie predpokladaných hodnôt so skutočnými hodnotami.
- Predikované várdy/X- hodnoty predpovedané modelom.
- Skutočné várdy/Y- skutočné hodnoty.
- Modré bodky-tréningové údaje množiny údajov.
- Zelené štvorce- testovacie údaje súboru údajov.
- Diagram ukazuje porovnanie medzi predpovedanými hodnotami a skutočnými hodnotami.

Kód na nastavenie:

```
svm_model = SVR(kernel='rbf', C=1000, epsilon=0.1,gamma=0.8)
svm_model.fit(X_train, y_train)
y_pred = svm_model.predict(X_test)
r2 = r2_score(y_test, y_pred)
```

Vyskúšala som niekoľko možností jadra, ako napríklad: lineárny, poly, sigmoid, rbf. Mne sa najviac osvedčilo rbf. Keďže som si vybrala toto jadro, nestačilo, že som musel nastaviť hodnotu C, ale musela som určiť aj gammu. Nebolo by potrebné nastavovať hodnotu epsilon. Jednotlivé hyper parametre majú tieto hodnoty, pretože po variáciách, ktoré som vyskúšal, sa mi zdali ako tie pravé.

Po niekoľkých pokusoch som si tiež uvedomil, že ak zmením hodnotu test_size na menšie číslo, napríklad 0,0015, môžeme dosiahnuť lepšiu hodnotu skóre R2. No nielen veľkosť testu ovplyvňuje aj túto hodnotu, ale aj hodnotu random_state. Iný výsledok dostaneme, ak je napríklad hodnota 42 a inú hodnotu môžem dosiahnuť, ak je hodnota náhodného stavu 45. Ak zároveň test_size=0,015, random_state=45, dostaneme nasledovné hodnoty:

```
SVM
R2 Score: 0.3948160169348428
************************************************
SVM Regression - Train
R2 Score: 0.26827941072655537
SVM Regression - Test
R2 Score: 0.3948160169348428
************************************************
SVM Regression - Train
Mean Squared Error: 256283162.03309467
Root Mean Squared Error: 16008.846367964641
R2 Score: 0.26827941072655537
SVM Regression - Test
Mean Squared Error: 106854799.15153593
Root Mean Squared Error: 10337.059502176426
R2 Score: 0.3948160169348428
```

Obrázok 24. Lepšie SVM

Porovnanie modely

	Decision tree	Bagging random forest	SVM
Train - R2 Score	0.806246873208491	0.8189824292671666	0.27511450988792074
Train - MSE	63804659.779644914	59610725.80353525	238711358.32337582
Train - RMSE	7987.781906114169	7720.798262066899	15450.286674472283
Test - R2 Score	0.5120121221058012	0.5955569501431877	0.13896113144695066
Test - MSE	249491990.78823322	206778295.60987204	440220569.38866013
Test - RMSE	15795.315469728144	14379.78774564743	20981.43392117565

Je potrebné preskúmať, ktorý model je najlepší z viacerých hľadísk. Na základe modelu R2 je pre nás jednoznačne najlepšou voľbou Random Forest, ktorý má najvyššiu hodnotu skóre R2 v Teste. Ďalším faktorom, na základe ktorého musíme jednotlivé modely skúmať, je MSE a RMSE. Aj tu sa najlepšie osvedčuje model Random forest, keďže čím menšie čísla pre MSE a RMSE, tým lepšie výsledky možno získať. Najmenšie hodnoty sa získavajú pre náhodný les, takže možno povedať, že najlepším modelom z hľadiska RMSE a MSE je Random Forest. Ak zhrnieme oba aspekty, najlepším modelom je Random Forest.

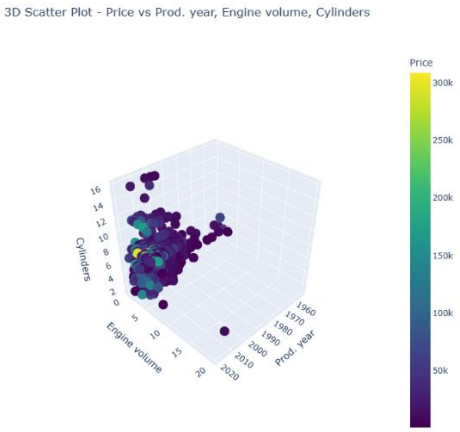
Redukcia dimenzie

Vybrané prínaky sú:

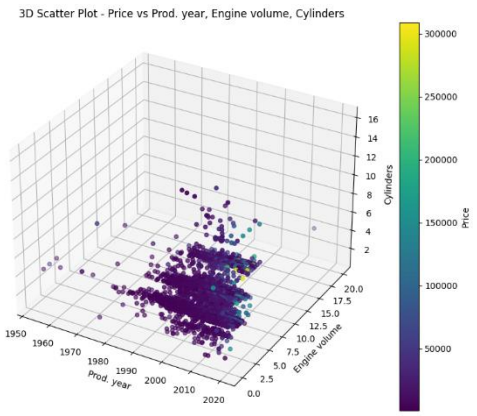
- Prod. Year
- Engine volume
- Cylinders

Hodnoty sú zafarbené podľa Price.

Pred normalizáciou:



Obrázok 26. 3D pred normalizáciou



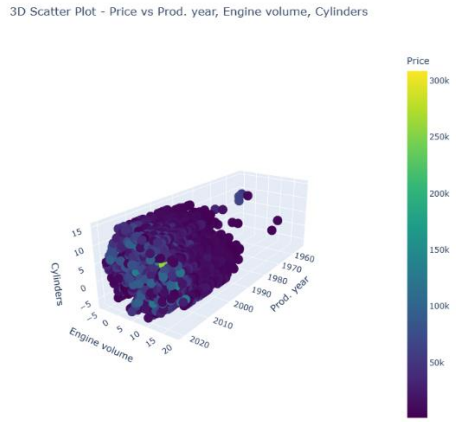
Obrázok 25. 3D pred normalizáciou

V našom 3D grafe jasne vidíme, že väčšina áut je z roku 2000. Motor sa zvyčajne skladá z 3-8 valcov, ale najbežnejší motor pozostáva zo 4 valcov. Výkon motora sa pohybuje od 1.3 do 2.5 podľa frekvencie. Dá sa tiež pozorovať, že najdrahšie autá sú tie z roku 2012.

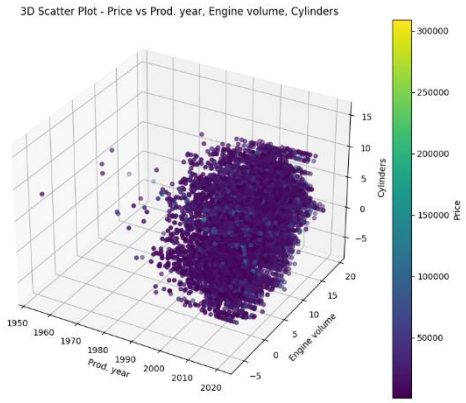
Pozorovateľným záverom je, že väčšina áut (z hľadiska frekvencie) v súbore údajov je z rokov 2000 až 2020, výkon motorov áut sa pohybuje od 1 do 3 a priemerný motor automobilu pozostáva z 3 až 8 valcov. Najdrahšie autá sú z rokov 2015, 2018, 2019. Všetky 3 autá majú 8-valcové motory. A výkon motora je 4, 3.4, 4.

Všetky 3 majú približne rovnaké vlastnosti.

Po normalizácií:



Obrázok 28. 3D po normalizácií



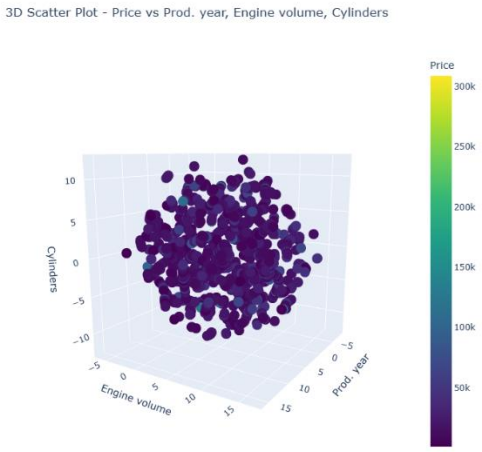
Obrázok 27. 3D po normalizácií

V tomto prípade som minimalizovala pomocou PCA, UMAP. Kód vyzerá nasledovne:

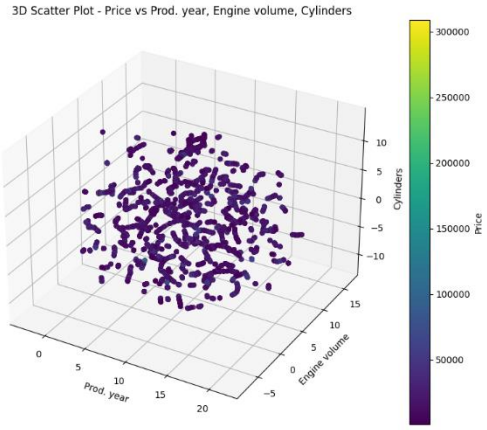
```
pca = PCA(n_components=3)
X_pca = pca.fit_transform(X)
umap_reducer = umap.UMAP(n_components=3)
X_umap = umap_reducer.fit_transform(X_pca)
reduced_df = pd.DataFrame(X_umap, columns=['Prod. year', 'Engine volume', 'Cylinders'])
reduced_df['Price'] = df['Price']
reduced_df['Prod. year'] = df['Prod. year']
```

Najprv vyskúšala som, že Prod. year nechala som tak ako bolo.

Na nasledujúcom obrázku by som chcela ukázať, že, ak Prod. year znížim ako bude vyzerať 3D graf. (oveľa inak)



Obrázok 30. Redukcia



Obrázok 29. Redukcia

Po viditeľnom zmenšení sme z jednotlivých údajov dostali pekný guľovitý tvar. Dĺžka x, y, z je úplne zmenená. Prod. rok sa teraz pohybuje od 0 do 20, objem motora sa pohybuje od -5 do 15, valce sa pohybujú od -10 do 10. Čo možno jasne pozorovať, je, že dĺžka každej osi je 20. Výrazne vyššie ceny sa nedajú nájsť. Podstatou UMAP je teda redukcia dát na menší priestor.

Podmnožina príznakov podľa koleračnej matice

Do súboru boli zahrnuté tie prvky, ktoré mali vysokú koreláciu. Keďže som chcela vysoko korelované položky, začína to na 0,7. Ale keďže vtedy bolo možné uvažovať len o dvoch prvkoch, zahrnul som do množiny aj prvky s koreláciou väčšou ako 0,65.

(correlation_values > 0.65) & (correlation_values < 1)

Engine volume	Cylinders	0.722110
Cylinders	Engine volume	0.722110
No	Yes	1.000000
Yes	No	1.000000
Automatic	Tiptronic	0.652725
Tiptronic	Automatic	0.652725

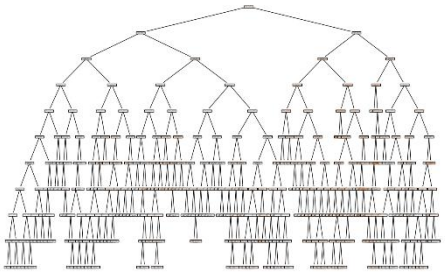
Obrázok 31. Kolerácia

Vybrala som teda tieto prvky:

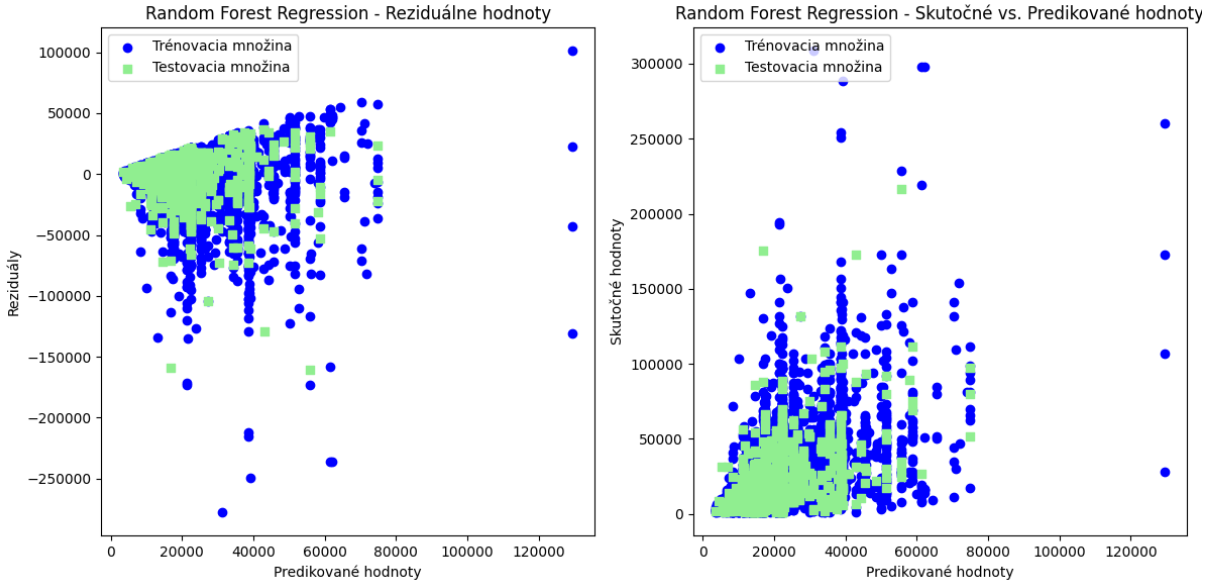
- Cylinders
- Engine volume
- Automatic
- Tiptronic
- Price (kvôli X, y)

```
Bagging random forest
R2 Score: 0.22190142451357053
*****************************************
Random Forest Regression - Train
Mean Squared Error: 272925049.14197433
Root Mean Squared Error: 16520.44337001808
R2 Score: 0.22131493372567324
Random Forest Regression - Test
Mean Squared Error: 250525602.0760802
Root Mean Squared Error: 15828.000571015917
R2 Score: 0.22190142451357053
```

Obrázok 33. Random forest podľa koleračnej matice



Obrázok 32. Strom

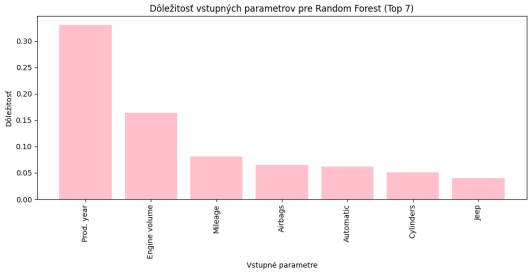


Obrázok 34. Reziduálne hodnoty

V tomto prípade je možné pozorovať, že som dostal oveľa menšiu hodnotu pre skóre R2 ako pre pôvodný náhodný les, avšak tieto dve skóre R2 sú teraz veľmi blízko. (Train-R2: 0.22131493372567324, Test-R2: 0.22190142451357053)

Podmnožina príznakov podľa dôležitosti príznakov z ensemble modelu

Podľa tejto grafy musela som vybrať príznaky:



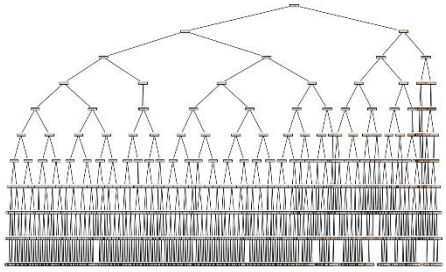
Obrázok 35. Príznaky podľa ensemble

Vybral som teda tieto prvky:

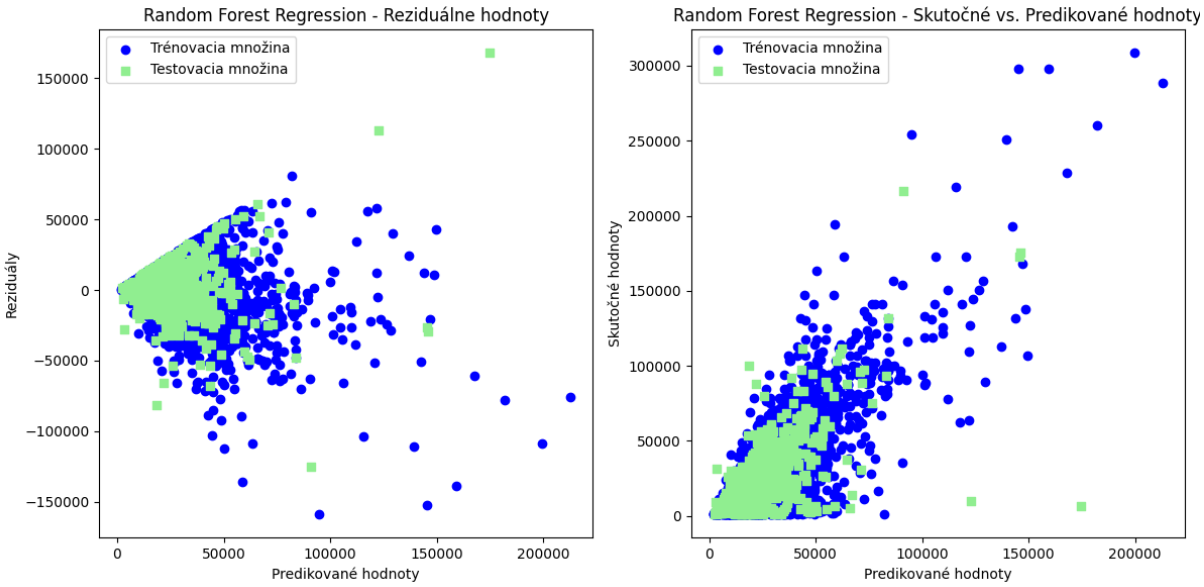
- Prod. Year
- Engine volume
- Mileage
- Price (kvôli X, y)

```
Bagging random forest
R2 Score: 0.4518009594353075
**************************************************
Random Forest Regression - Train
Mean Squared Error: 131516999.16372594
Root Mean Squared Error: 11468.0861159884
R2 Score: 0.6247675926707148
Random Forest Regression - Test
Mean Squared Error: 176504493.16031983
Root Mean Squared Error: 13285.499356829605
R2 Score: 0.4518009594353075
```

Obrázok 37. Random forest podľa ensemble



Obrázok 36. Strom



Obrázok 38. Reziduálne hodnoty

V tomto prípade je hodnota skóre R2 lepšia ako predchádzajúca, ale stále nie lepšia ako pôvodná.

Podmnožina príznakov podľa PCA

Vybral som príslušné prvky pomocou nasledujúcej časti kódu:

```
pca = PCA(n_components=3)
X_scaled = StandardScaler().fit_transform(X)
pca.fit(X_scaled)
explained_variance = pca.explained_variance_
components = pca.components_
```

```
component_weights = np.abs(components)
selected_features = [X.columns[i] for i in range(len(X.columns)) if any(component_weights[:, i] > 0.1)]
```

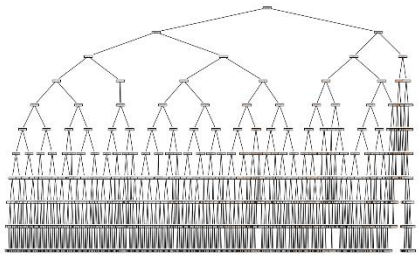
Podarilo sa mi vybrať nasledujúce hodnoty:

['Price','Prod. year', 'Engine volume', 'Cylinders', 'Airbags', 'BMW', 'HONDA', 'HYUNDAI', 'MERCEDES-BENZ', 'OPEL', 'SSANGYONG', 'TOYOTA', 'Coupe', 'Goods wagon', 'Hatchback', 'Jeep', 'Microbus', 'Minivan', 'Sedan', 'CNG', 'Diesel', 'Hybrid', 'Petrol', 'Automatic', 'Manual', 'Tiptronic', 'Variator']

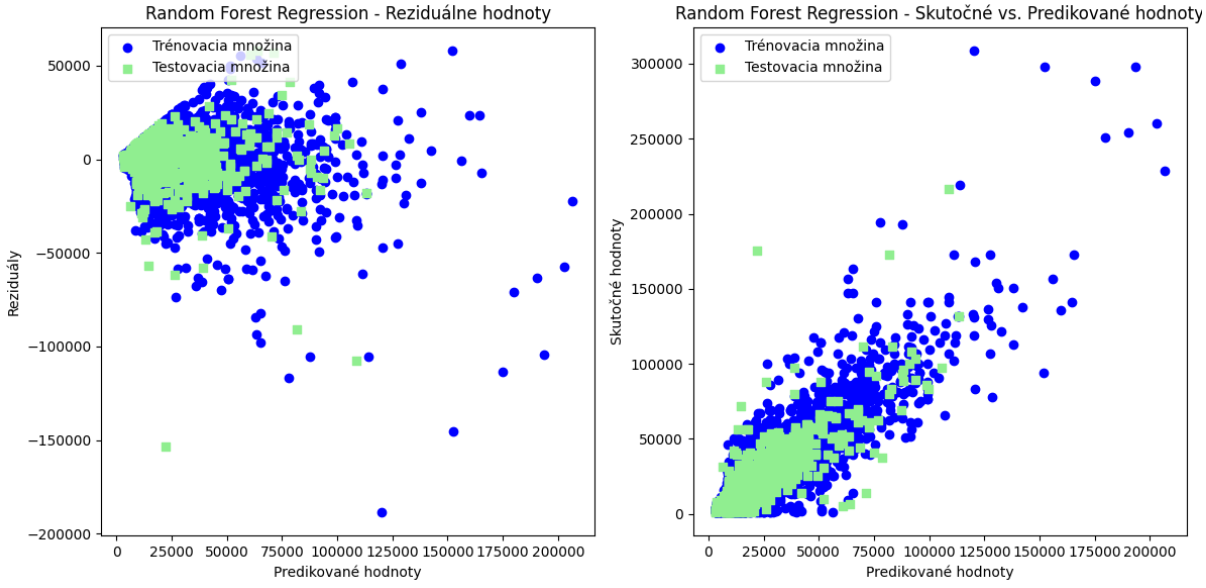
A dosiahnuté výsledky pomocou tohto priebehu vyzerajú takto:

```
Bagging random forest
R2 Score: 0.6890745844935997
**************************************************
Random Forest Regression - Train
Mean Squared Error: 76313402.82361127
Root Mean Squared Error: 8735.754279031164
R2 Score: 0.7822695010145038
Random Forest Regression - Test
Mean Squared Error: 100109137.0354975
Root Mean Squared Error: 10005.455363725207
R2 Score: 0.6890745844935997
```

Obrázok 40. Random forest podľa PCA



Obrázok 39. Strom



Obrázok 41. Reziduálne hodnoty

Ak zhrnieme doterajšie Random Forest, s týmto modelom som získala najlepšie hodnoty pre skóre R2.

Porovnanie Random Forest

	Bagging random forest	RF podľa koleračnej matice	RF podľa ensemble	RF podľa PCA
Train - R2 Score	0.8189824292671666	0.22131493372567324	0.6247675926707148	0.78226950101450
Train - MSE	59610725.80353525	272925049.14197433	131516999.16372594	76313402.8236112
Train - RMSE	7720.798262066899	16520.44337001808	11468.0861159884	8735.75427903116
Test - R2 Score	0.5955569501431877	0.22190142451357053	0.4518009594353075	0.68907458449359
Test - MSE	206778295.60987204	250525602.0760802	176504493.16031983	100109137.035497
Test - RMSE	14379.78774564743	15828.000571015917	13285.499356829605	10005.4553637252

Model 4 (podľa PCA) má najvyššie R2 score.

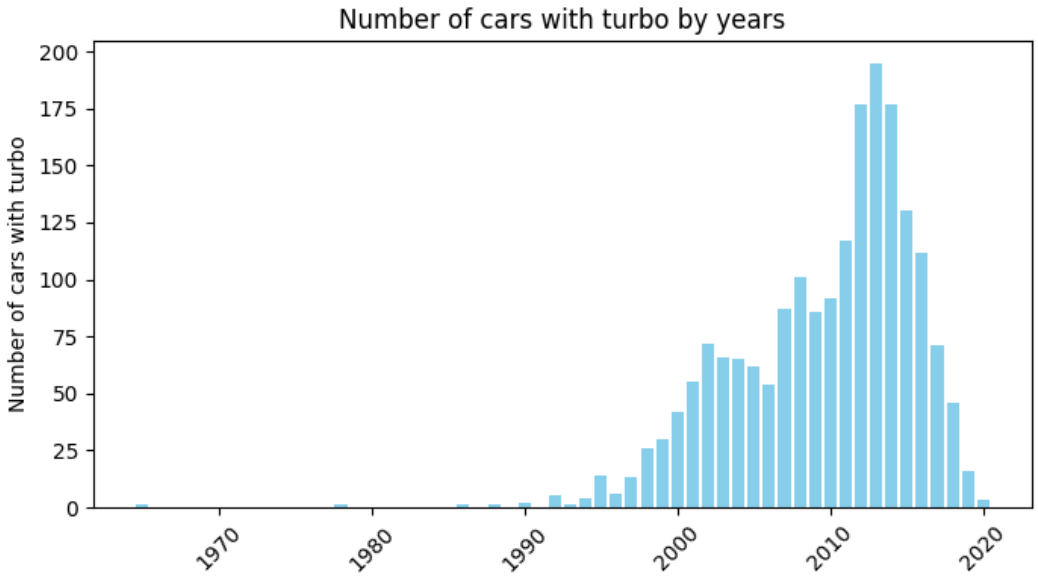
Model 4 (podľa PCA) má najnižšiu MSE, čo znamená, že hodnoty odhadované modelom sú bližšie k skutočným hodnotám. Nižšie MSE znamená lepší prediktívny výkon.

Teda ohľadom R2 score aj ohľadom MSE a RMSE má najlepšie hodnoty.

Bonusová úloha: EDA

EDA 1

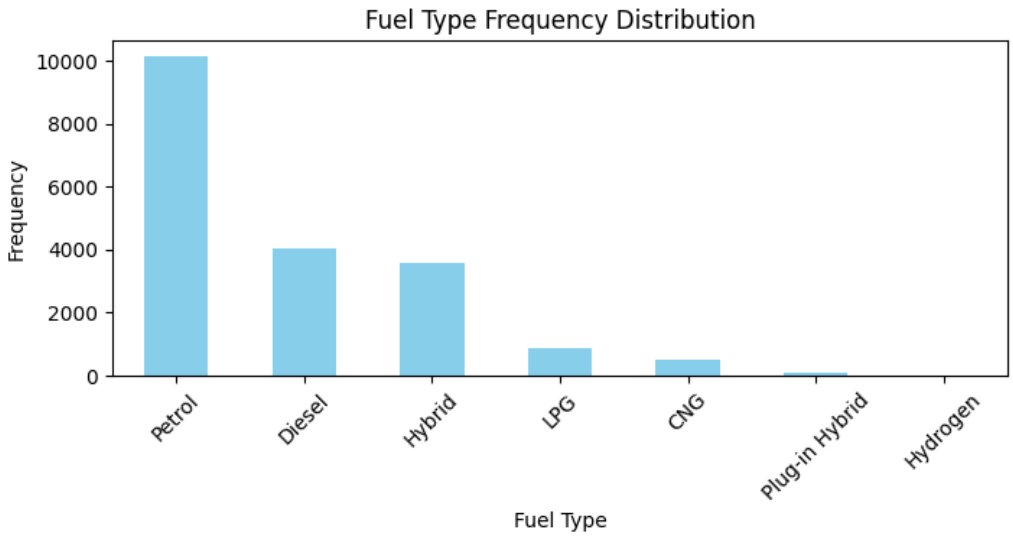
Na tomto grafe môžeme sledovať, koľko áut s turbami bolo vyrobených v ktorom roku, čo je zahrnuté aj v našom súbore údajov. Z obrázku je zrejmé, že najviac áut s turbami bolo v roku 2013, no v rokoch 2012 a 2014 bolo prihlásených aj dosť áut s turbami. Ide o typu bar EDA.



Obrázok 42. EDA 1

EDA 2

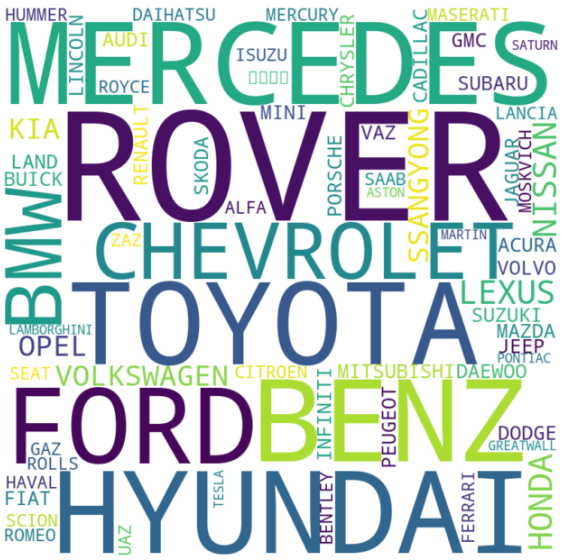
Na tomto EDA môžete vidieť, ktorý typ paliva je najbežnejší. Mimoriadne veľa áut jazdí na benzín, ale aj to sa dalo očakávať. Druhým najbežnejším palivom je nafta. V databáze je veľmi málo áut fungujúcich v Hybridnom režime, no taktiež takmer žiadne autá na vodík. Toto je jednoduchá plot EDA.



Obrázok 43. EDA 2

EDA 3

Posledná EDA je jedna z najzaujímavejších. Ukazuje výskyt automobiliek. Čím väčšie slovo, tým viac ho nájdete v kolónke Výrobca. Je vidieť, že najbežnejšie autá sú Mercedes, Rover, Toyota, Ford, Benz, BMW, Hyundai. Toto je WordCloud EDA.



Obrázok 44. EDA 3