

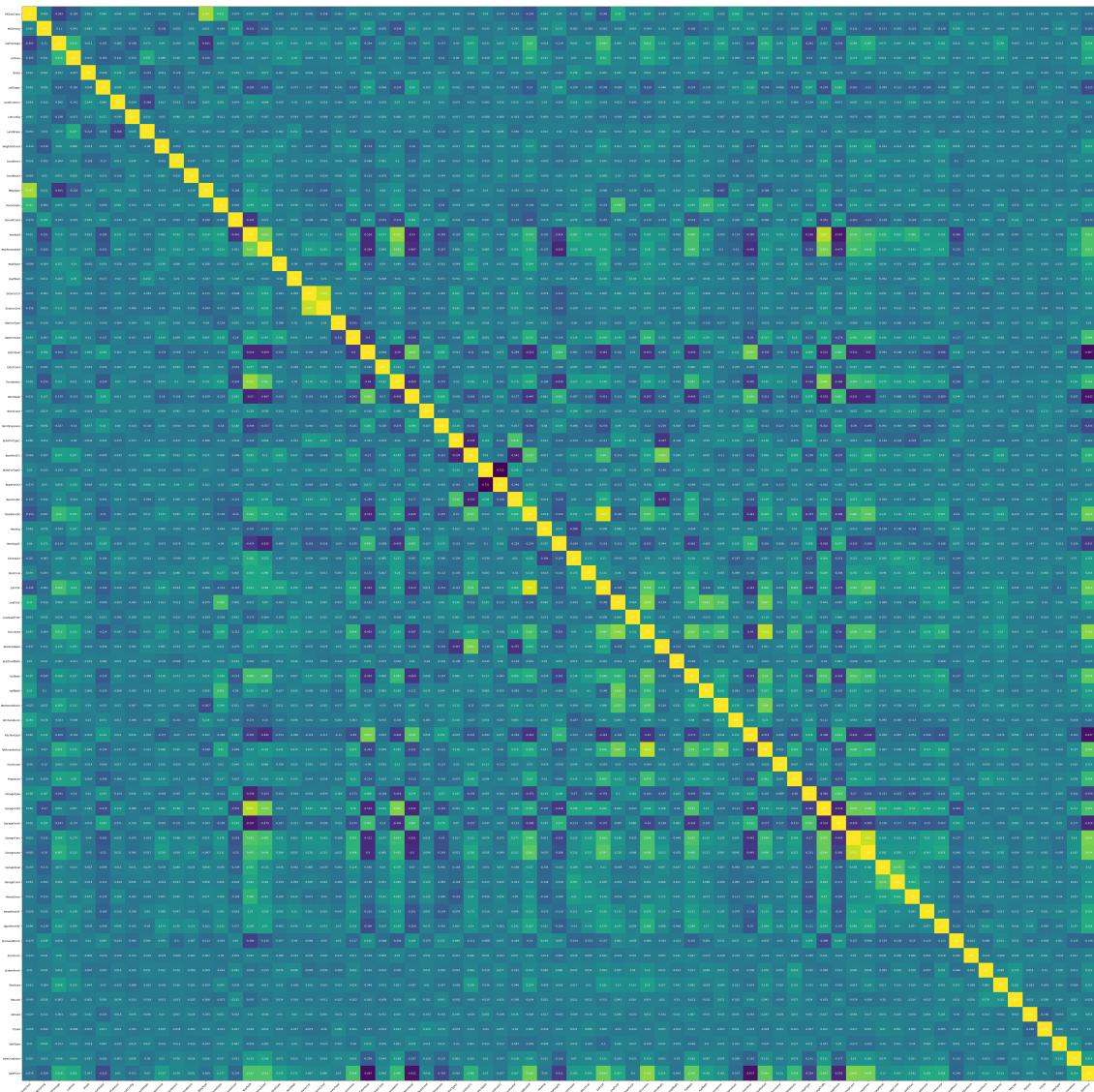
SUNS - Zadanie 2

28. novembra 2022

1 Povinne úlohy

1.1 Spracovanie dát

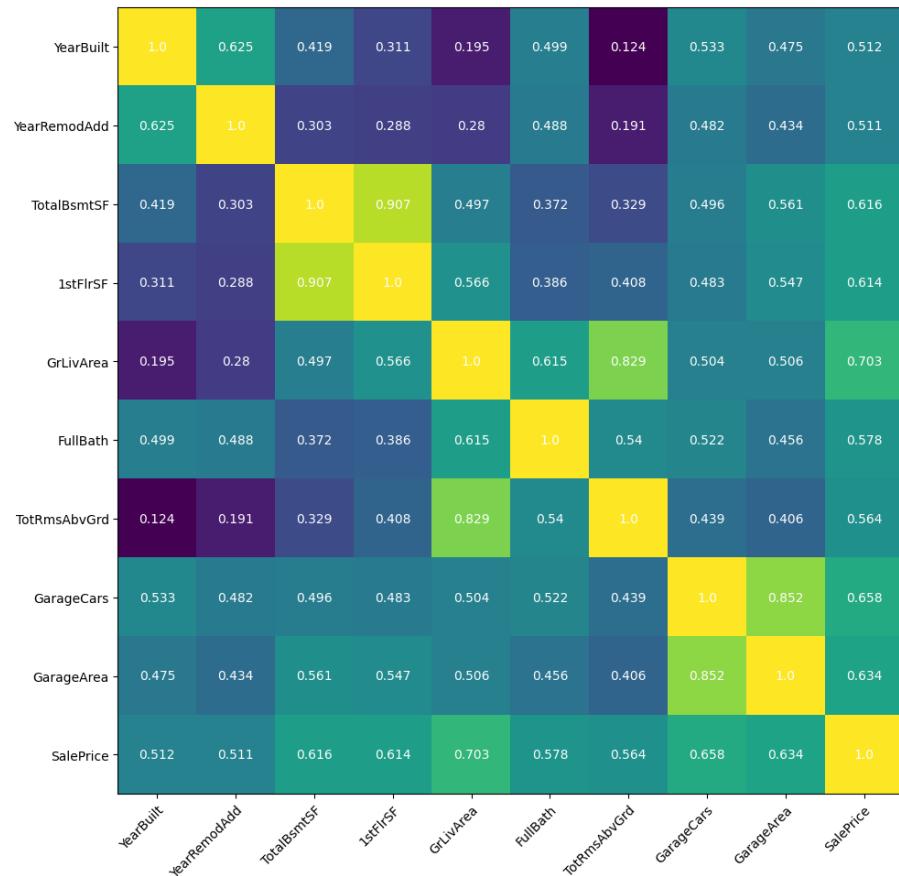
Použil som knižnicu **Pandas** na načítanie a spracovanie dát a pracoval som s **DataFrame** štruktúrou. Zbavil som sa všetkých nulových a nezmyselných hodnôt a konci som normalizoval dáta. Najprv som sa pozrel na **korelačnú maticu** celého DataFrame. Vidíme ju na obrázku 1.



Obrázok 1: Korelačná matica všetkých dát

Potom som na prípravu dát pre testovanie zmazal stĺpce, ktoré mali najmenšiu koreláciu s naším testovacím stĺpcom sale_price. Dôsledok toho je korelačná matica, ktorá bližšie koreluje so stĺpcom sale_price (obrázok 2). Tieto stĺpce sme využili vo trenovacej a

testovacej množine.



Obrázok 2: Korelačná matica upravených dát

1.2 Trenovanie

Na trenovanie som použil **GridSearch** pre 3 rôzne modely:

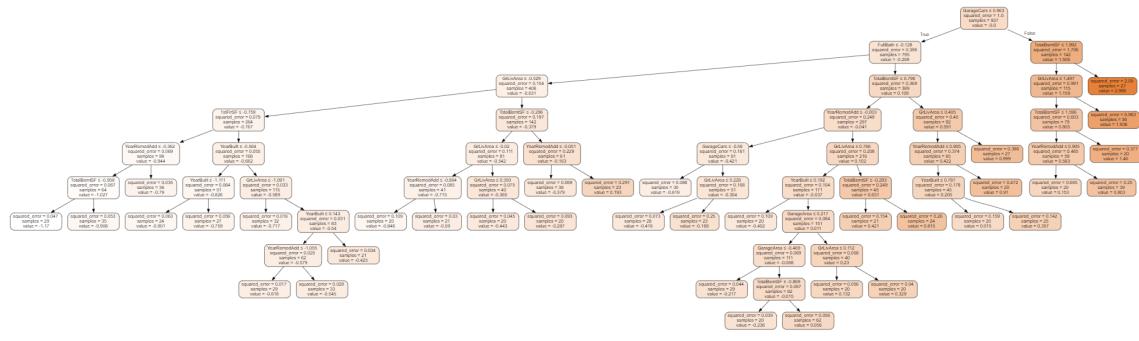
- Decision Tree
- Support Vector Regressor
- Random Forest

Pri každom modely sme trénovali sieť nad hyperparametrami a cross validáciou.

1.2.1 Decision Tree

Decision Tree je jeden zo najznámejších algoritmov na trenovanie modelov, a kvôli regresnému problému sme využili DecisionTreeRegressor(). Tento algoritmus vytvára nové predpovede na základe predošlých. **R2 score** pre túto sieť bol **0.664**, čo je dobre natréno-

vaný modél. Na obrázku 3 vidíme výsledný strom a na obrázku 4 vidíme parametre ktoré sme použili.



Obrázok 3: Rozhodovací strom

```

from sklearn import tree
from sklearn.model_selection import GridSearchCV

dec_tree = tree.DecisionTreeRegressor(random_state = 0)

param_grid = {
    'criterion': ['squared_error'],
    'min_samples_split': range(2, 10),
    'max_depth': range(2, 10),
    'min_samples_leaf': [20, 50, 100]
}

clf_gs = GridSearchCV(dec_tree, param_grid = param_grid, scoring = 'r2', cv = 3)
clf_gs.fit(X, y)
print()

```

```

# on the test or hold-out set
from sklearn.metrics import r2_score
print("R2 score: ", clf_gs.score(X_test, y_test.ravel()))

best_tree = clf_gs.best_estimator_

print('Best Parameters:', clf_gs.best_estimator_.get_params())

R2 score:  0.664551047577958

```

Obrázok 4: Výsledky rozhodovacieho stromu

1.2.2 Support Vector Regressor

Support Vector Regressor (**SVR**) je iný typ modelu na trenovanie sieti, ktorý rieši regresny problém pomocou vektorov. Vyžaduje kernel, hodnotu C parametra a hodnotu gamma parametra. Ja som si zvolil následné hyperparametre a dostál som **R2 skóre** na testovacej množine **0.84**, čo je vynikajúca hodnota (obrázok 5).

```
svr_reg_test = model.predict(x_test)
print("SVR Test MSE: %.2f"
      % np.sqrt(mean_squared_error(y_test, svr_reg_test)))
print("SVR Test R^2 Score: %.2f"
      % r2_score(y_test, svr_reg_test))

svr_eval(svr, x, y, x_test, y_test)
best_model_svr = svr.best_estimator_

SVR Train MSE: 0.36
SVR Train R^2 Score: 0.87
SVR Test MSE: 0.40
SVR Test R^2 Score: 0.84

svr = GridSearchCV(
    SVR(),
    param_grid = {"kernel": ['rbf'], "C": [1e0, 1e1, 1e2, 1e3], "gamma": [0.01, 0.1, 1]},
    cv = 3
)
svr.fit(x, y.ravel())
```

Obrázok 5: Support Vector Regressor výsledky

Následne, použil som iný typ kernela, a som dostál trochu horšie, ale podstačujúce výsledky (obrázok 6).

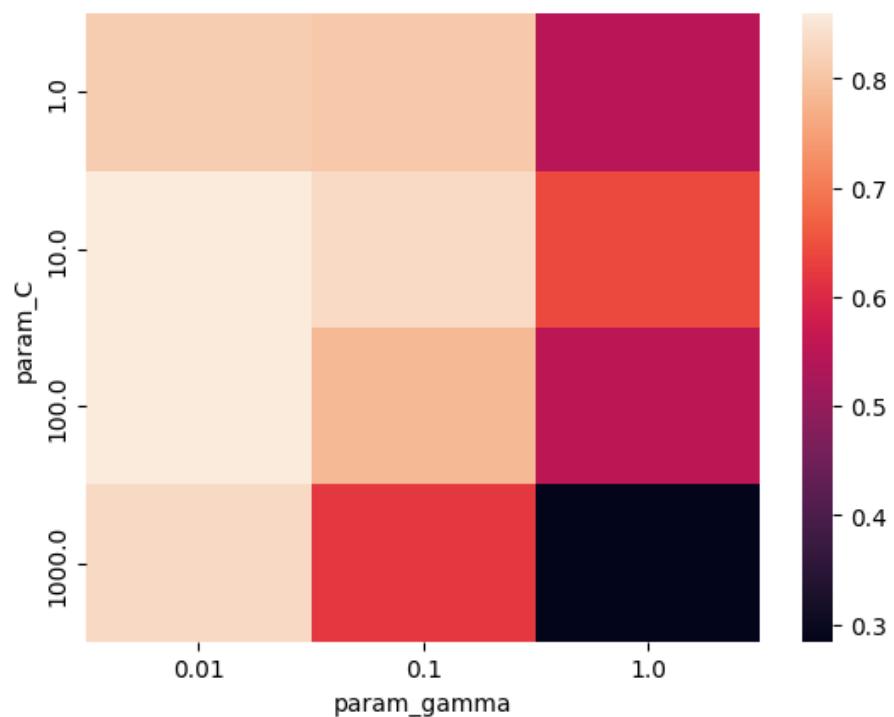
```
svr_2 = GridSearchCV(
    SVR(),
    param_grid = {"kernel": ['sigmoid'], "C": [1e0, 1e1, 1e2, 1e3], "gamma": [0.01, 0.1, 1]},
    cv = 3
)
svr_2.fit(x, y.ravel())

svr_eval(svr_2, x, y, x_test, y_test)

SVR Train MSE: 0.50
SVR Train R^2 Score: 0.75
SVR Test MSE: 0.49
SVR Test R^2 Score: 0.76
```

Obrázok 6: Support Vector Regressor výsledky s iným kernelom

Prvý modél mal krajšie výsledky, tak som pre tú množinu ešte vytvoril heatmap pre C a gamma hodnoty (obrázok 7).



Obrázok 7: Heatmap

1.2.3 Random Forest Regressor

Ako môjim vybraným súborovým modéjom som si zvolil `RandomForestRegressor()`. S ním som dostál dobré výsledky; **r2 skóre** je **0.77** (obrázok 8).

Taktiež sme vizualizovali dôležitosť vstupných parametrov (obrázok 9).

```

from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import RandomForestRegressor

n_estimators = [5, 20, 50, 100]
max_features = ['auto', 'sqrt']
max_depth = [int(X) for X in np.linspace(10, 120, num = 12)]
min_samples_split = [2, 6, 10]
min_samples_leaf = [1, 3, 4]
bootstrap = [True, False]

random_grid = {
    'n_estimators': n_estimators,
    'max_features': max_features,
    'max_depth': max_depth,
    'min_samples_split': min_samples_split,
    'min_samples_leaf': min_samples_leaf,
    'bootstrap': bootstrap
}

rf = RandomForestRegressor()
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,
                                n_iter = 100, cv = 5, verbose = 2, random_state = 35, n_jobs = -1)

rf_random.fit(X, y)
print()

```

```

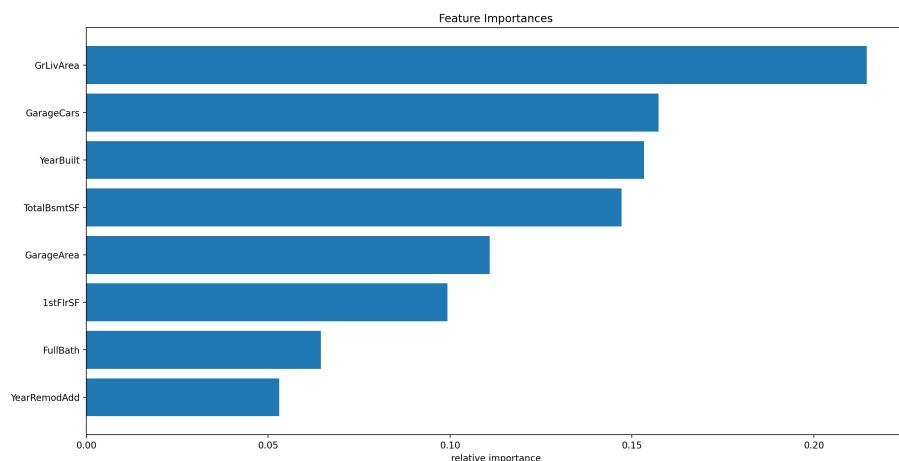
# print('Random grid: ', random_grid, '\n')
# print the best parameters
print('Best Parameters: ', rf_random.best_params_, '\n')
print("r2score:", rf_random.score(X_test, y_test))

best_random = rf_random.best_params_

Best Parameters: {'n_estimators': 100, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'sqrt', 'max_depth': 10, 'bootstrap': False}
r2score: 0.7707039208759212

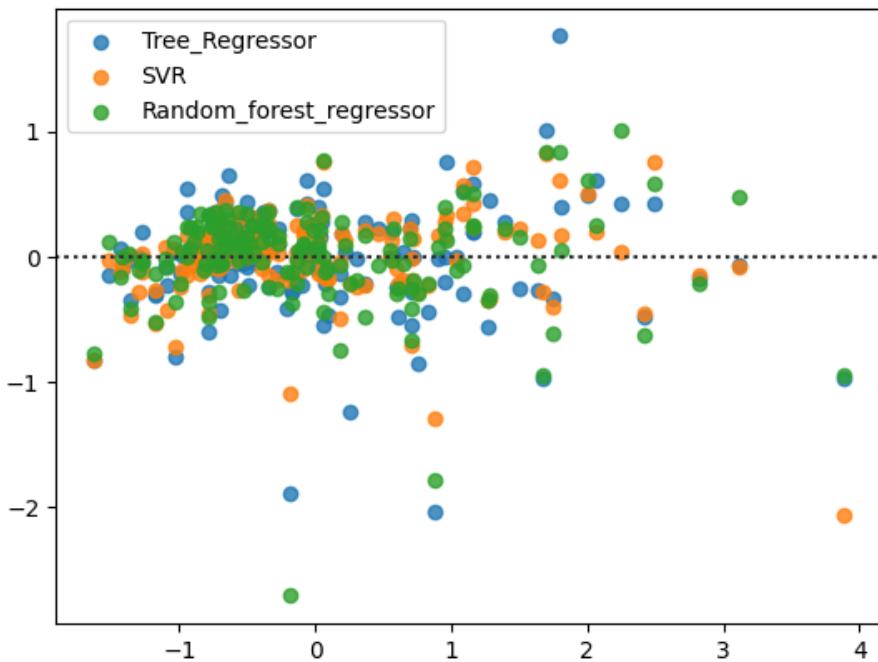
```

Obrázok 8: Random Forest Regressor



Obrázok 9: Graf trenovania

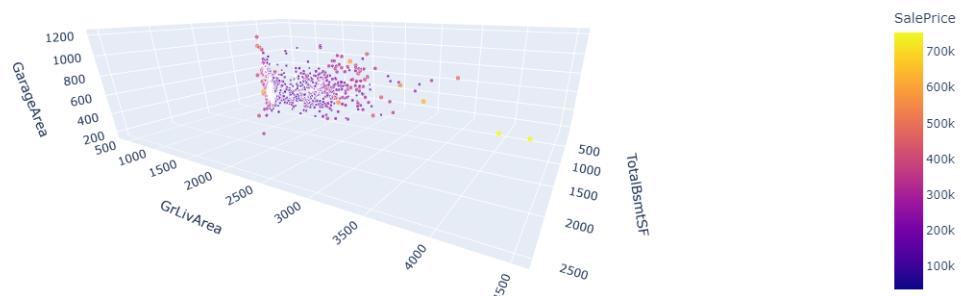
Nakonci sme porovnali všetky 3 modeli pomocou reziduálov. Výsledky sú podobné, keďže sú r² skóre podobné (obrázok 10).



Obrázok 10: Reziduály

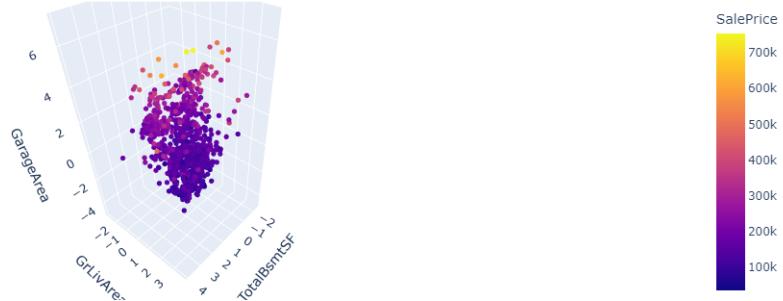
1.3 Redukcia dimenzie

Na obrázku 11 vidíme znázornenie našej databázu na základe 3 príznakov v 3D priestore. Príznaky sme si vybrali na základe feature importances a korelačnej matice. Dáta sme vyfarbili podľa vystupného parametra 'SalePrice'.



Obrázok 11: 3D

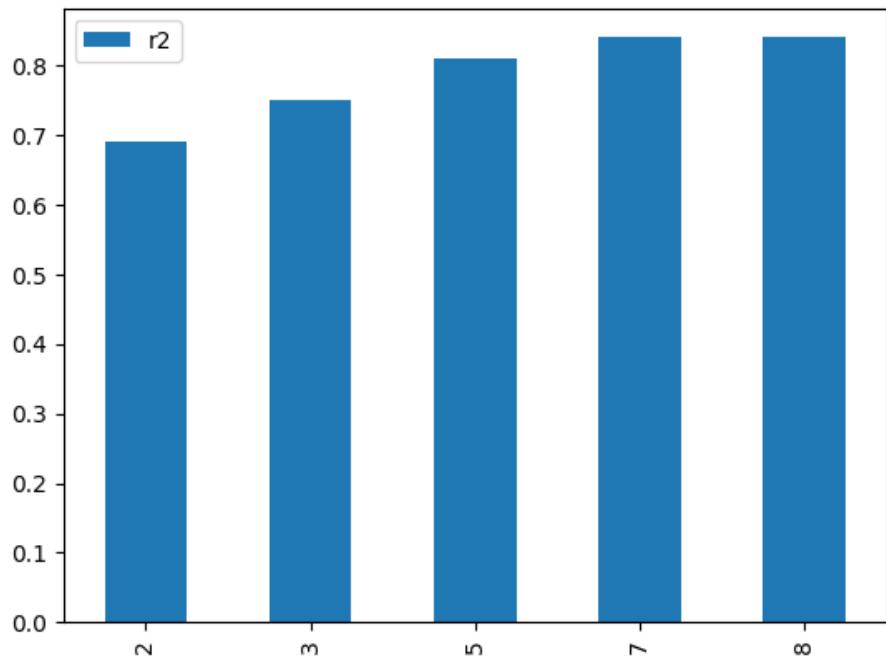
Túto množinu sme potom zredukovali pomocou PCA(). (obrázok 12).



Obrázok 12: Zredukovaná množina

1.4 Redukcia podmnožiny na X dimenzií

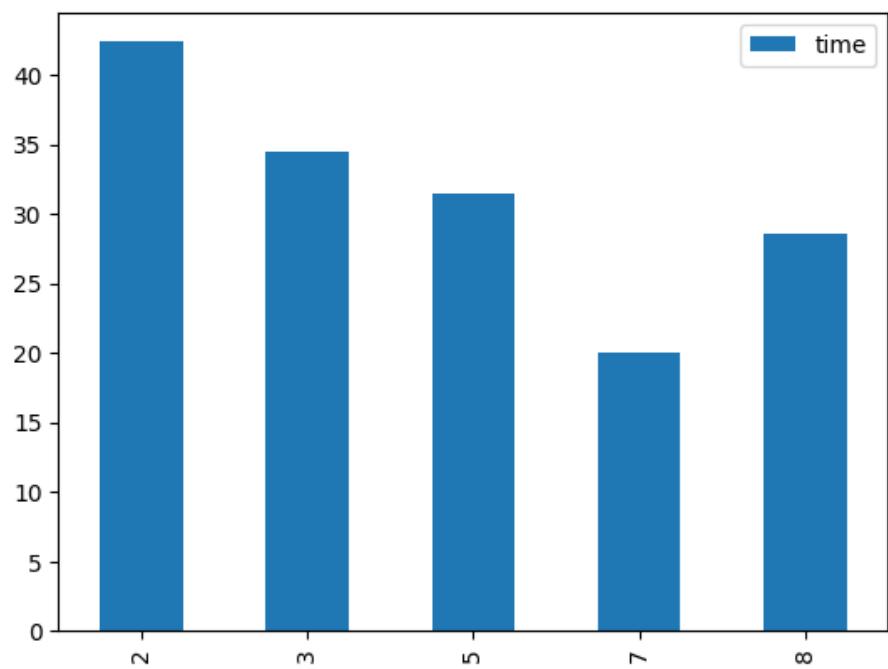
V tejto úlohe som mal splniť viacero vecí. Množinu príznakov som si zvolil, že mi bude množina feature importances (obrázok 9). Na danú množinu som aplikoval rôzne veľkosti redukcie dimenzii (na 2, 3, 5, 7, 8). A na konci, som trénoval zmenšené množiny pomocou SVR(), lebo mi dával najlepšie výsledky. Výsledok (r^2 skóre) každého modélu som vyplotil na graf (obrázok 13).



Obrázok 13: R2 score pre každú zredukovanú množinu

Taktiež, som vyplotil aj čas trenovania koľko sekúnd bolo potrebné na trenovanie

každej množine (obrázok 14).

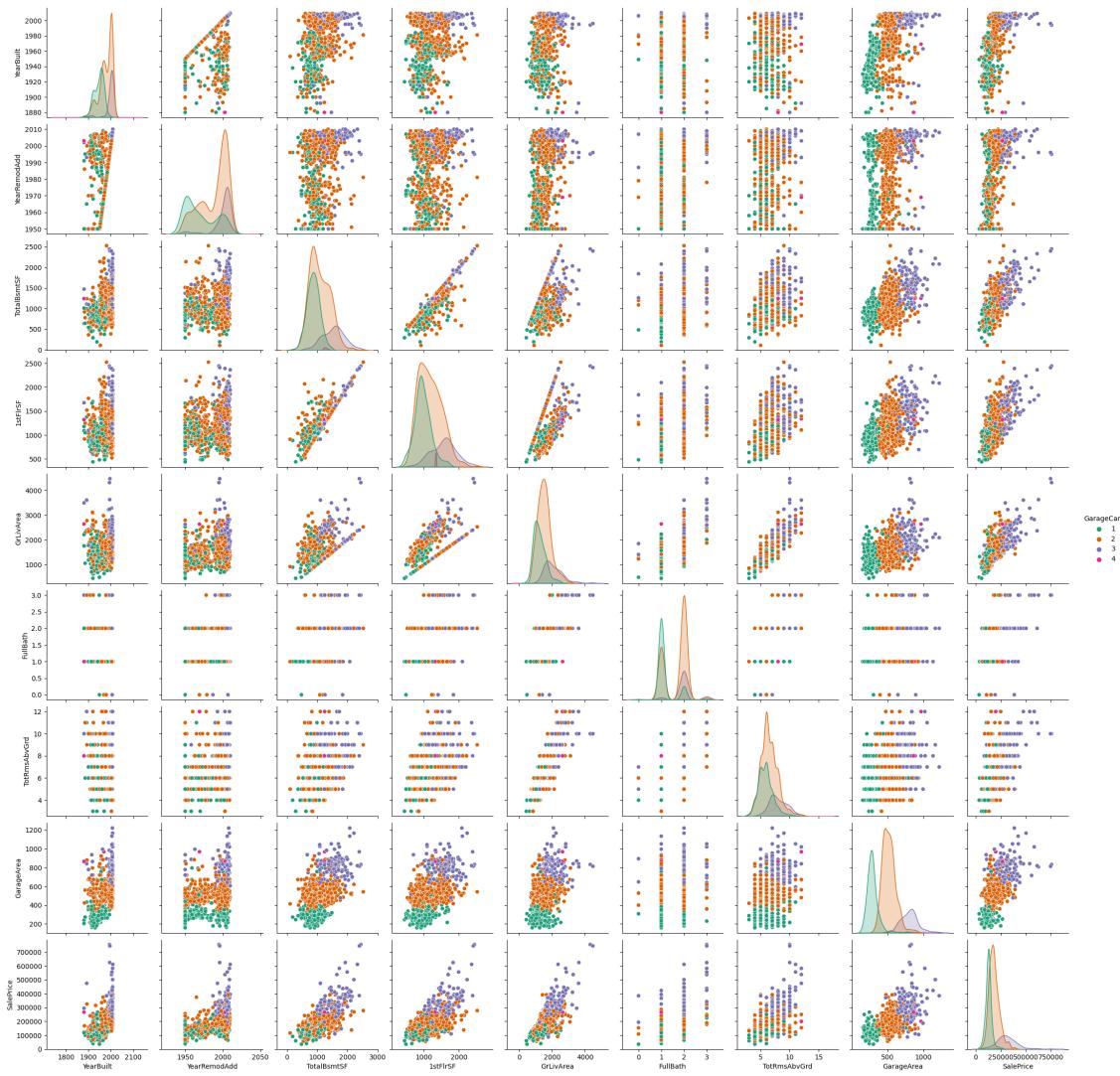


Obrázok 14: Čas trenovania (s) pre každú zredukovanú množinu

2 Nepovinné úlohy

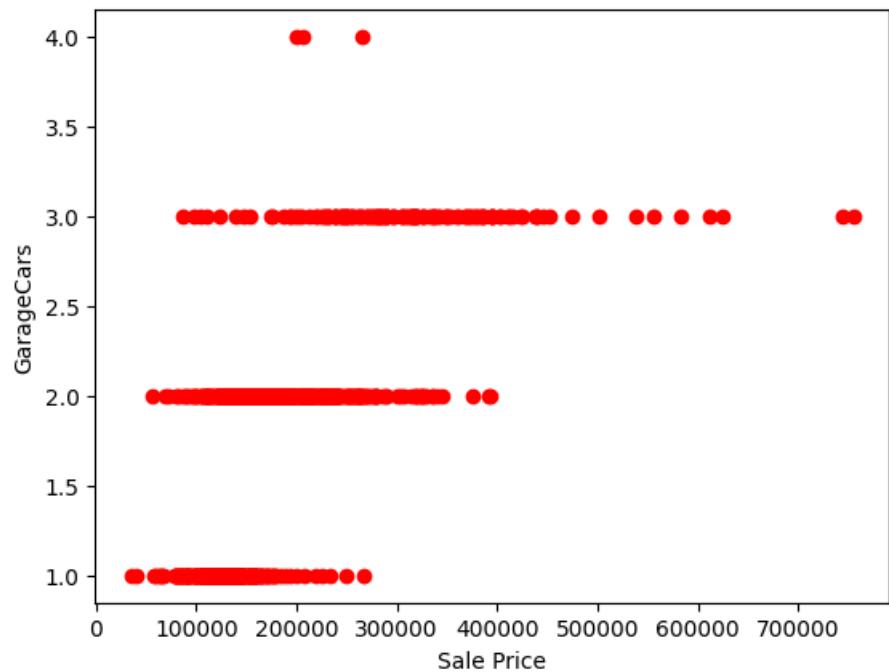
2.1 EDA

Pomocou EDA som si všimol niektoré hodnoty GrLivArea a TotalBsmtSF boli príliž nezmyselné, a som ich zmazal. Na to som získal lepšie výsledky pri trenovaní. Následne som znázornil koreláciu daných príznakov každý s každým a som ofarbil grafy pomocou hodnoty 'GarageCars'.



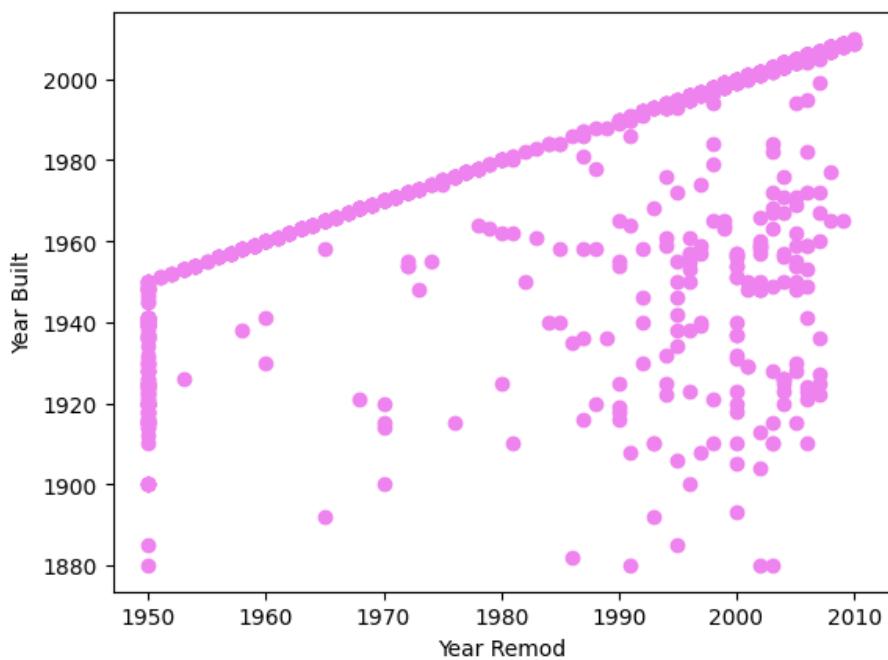
Obrázok 15: Snairplot

Aby tento graf bol prehľadnejší, zvolil som si niekoľko grafov na z ponuknútých na analýzu. Ako napríklad ako hodnota SalePrice rastie počtom hodnotou GarageCars.



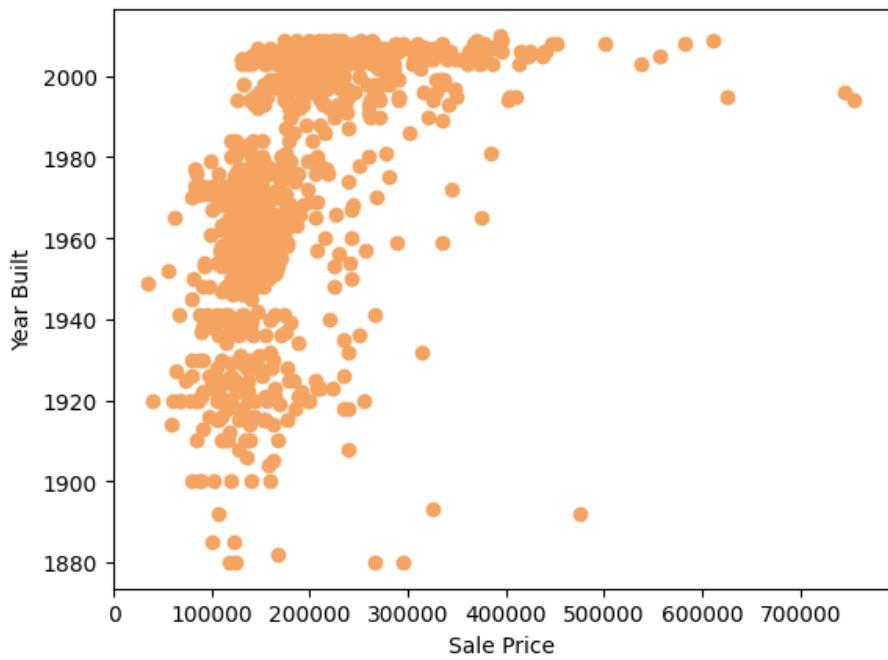
Obrázok 16: Korelácia SalePrice s GarageCars

Na tomto grafe je zaujímavý tvar gulôčiek. Veľa hypotékov mali datí remod v tom roku keď sa hypotéka vybudovala.



Obrázok 17: Korelácia YearBuilt s YearRemodAdd

Podobne ako pri prvom grafe, možné je vidieť rast v hodnote SalePrice na základe roku YearBuilt.



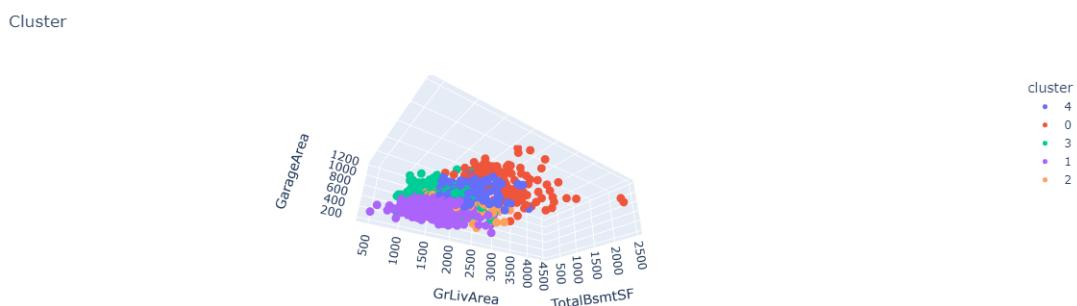
Obrázok 18: Korelácia SalePrice s YearBuilt

Ako na záver k EDA patrí aj spracovanie a úprava dátov množine, ktorú sme si

spomenuli na začiatku dokumentácie.

2.2 Zhlukovanie dát

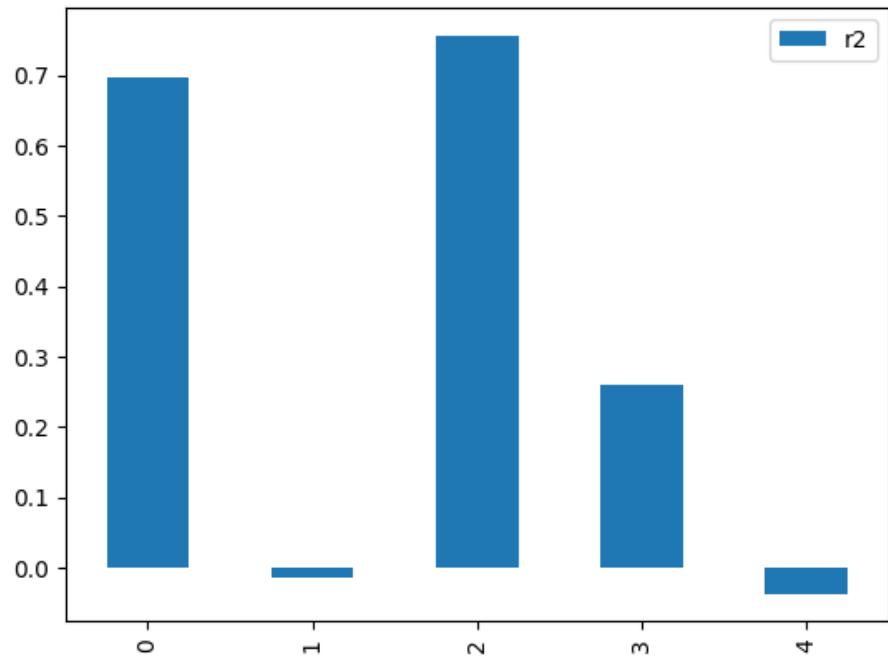
Zhlukovanie dát sme riešili pomocou metódou clustering. Využili sme naň KMeans(). Toto nám rozdelilo našu množinu na 5 rôznych častí, ako je možné vidieť na grafe 19. Niektoré množiny sú vynikajúco roztriedené a niektoré sú nie. V následovnej časti uvidíme, že niektoré roztriedené dáta sa ozaj dajú natrenovať.



Obrázok 19: Zhlukované dáta

2.3 Trenovanie zhlukovaných dát

Potom sme dáta rozdelili na 5 častí podľa clusterov. Po zhlukovaní sme dané množiny trenovali samostatne pomocou SVR(). Na obrázku 20 je znázornené R2 score pre každý cluster.



Obrázok 20: R2 score pre každý cluster