

Strojové učenie

(riešenie praktického problému pomocou algoritmov strojového učenia)

Riešiteľ: **Matúš Revický 1Im**, Ústav informatiky PF UPJŠ

Rok: 2020/2021 letný semester

Riešený problém:

Úlohou projektu je skonštruovať klasifikátor (vo forme rozhodovacieho stromu, booleovskej formuly, rozhodovacieho zoznamu, neurónovej siete a pod.) na základe tréningovej množiny.

Popis dát:

V úvode projektu bolo nevyhnutné nájsť si dataset, ktorý je zameraný na COVID-19, je ho možné rozdeliť na tréningovú a testovaciu množinu.

Rozhodol som sa pre dataset z portálu Kaggle s názvom COVID-19 Healthy Diet Dataset dostupný na <https://www.kaggle.com/mariaren/covid19-healthy-diet-dataset>. Tento dataset je licencovaný pod licenciou <https://creativecommons.org/licenses/by-sa/3.0/>.

V tomto datasete sa nachádzajú kombinované údaje o rôznych druhoch potravín, o miere obezity a podvýživy z celého sveta, počtoch ľudí s ochorením COVID-19. Z tohto datasetu by malo byť možné zistiť ako by štýl zdravého stravovania mohol pomôcť v boji proti COVID-19. Z datasetu môžeme vyčítať informácie týkajúce sa stravovacích návykov z krajín s nižšou mierou infekcie COVID a podľa toho upravovať vlastné stravovacie návyky.

Dataset pozostáva zo 4 csv súborov. Pre jednotlivé kategórie potravín je vypočítané množstvo tuku, energetický príjem (kcal), kvantita potravy (kg) a množstvo proteínov (všetky vypočítané ako percento z celkového množstva). V súboroch sa nachádzajú aj informácie o miere obezity a podvýživy vzhľadom na celkovú populáciu krajiny a taktiež sa nachádzajú aj informácie o percentuálnom podiele potvrdených (confirmed), vyliečených (recovered), aktívne chorých (active cases) a úmrtiach (deaths) na ochorenie COVID-19.

	Alcoholic Beverages	Animal Products	Animal fats	Aquatic Products, Other	Cereals- Excluding Beer	...
Fat_Supply_Quantity_Data.csv						
Afghanistan	0	21.6397	6.2224	0	8.0353	...
Protein_Supply_Quantity_Data.csv						
Afghanistan	0	9.7523	0.0277	0	35.9771	...
Food_Supply_Quantity_kg_Data.csv						
Afghanistan	0.0014	0.1973	9.4341	0	24.8097	...
Food_Supply_kcal_Data.csv						
Afghanistan	0	4.7774	0.8504	0	37.1186	...

Tab. 1 Ukážka častí jednotlivých tabuliek

Súčet hodnôt v jednotlivých riadkoch, ktoré prislúchajú atribútom týkajúcich sa potravín je 100. Všetky možné potraviny sú zaradené do 23 kategórií. Napr. v Afganistane

osoba zo všetkých tukov, ktoré prijíma je cca. 21% z živočíšnych produktov (Animal products). Zo všetkých proteínov, ktoré prijíma je cca. 9% z živočíšnych produktov. Priemerná osoba v Afganistane z celkovej hmotnosti potravy, ktorú prijíma je cca. 0.2% z živočíšnych produktov.

Údaje o rôznych množstvách jednotlivých skupín potravín, výživových hodnotách, obezite a podvyživených sú získané z Organizácie pre výživu a poľnohospodárstvo na webovej stránke Organizácie Spojených národov FAO¹. Konkrétne druhy potravín zahrnutých v každej kategórii z údajov FAO, sú dostupné v súbore Supply_Food_Data_Description.csv. Údaje o počte obyvateľov pre každú krajinu pochádzajú z webovej stránky Population Reference Bureau PRB². Údaje potvrdených (confirmed), vyliečených (recovered), aktívne chorých (active cases) a úmrtiach (deaths) na ochorenie COVID-19 sú získané z webových stránok CSSE Center for Systems Science and Engineering Johns Hopkins³.

Všetky 4 tabuľky majú rozmer (170, 32), pričom riadky reprezentujú 170 rôznych krajín sveta. Všetky atribúty okrem unit a country sú numerické. Atribút unit obsahuje vo všetkých prípadoch iba znak '%'. V stĺpcoch sú nasledujúce hodnoty:

Všetky vstupné premenné:

- | | |
|-----------------------------|----------------------------------|
| 1. Country | 17. Starchy Roots |
| 2. Alcoholic Beverages | 18. Stimulants |
| 3. Animal Products | 19. Sugar Crops |
| 4. Animal fats | 20. Sugar & Sweeteners |
| 5. Aquatic Products, Other | 21. Treenuts |
| 6. Cereals - Excluding Beer | 22. Vegetal Products |
| 7. Eggs | 23. Vegetable Oils |
| 8. Fish, Seafood | 24. Vegetables |
| 9. Fruits - Excluding Wine | 25. Obesity |
| 10. Meat | 26. Undernourished |
| 11. Milk - Excluding Butter | 27. Confirmed |
| 12. Miscellaneous | 28. Deaths |
| 13. Offals | 29. Recovered |
| 14. Oilcrops | 30. Active |
| 15. Pulses | 31. Population |
| 16. Spices | 32. Unit (all except Population) |

Cieľová premenná (na základe zmyslových dát):

Umelo vytvorený nominálny atribút s hodnotou **vyliečení (Recovered)/celkový počet diagnostikovaných (Confirmed)**, pričom výsledky budú vhodne rozložené do rozumného počtu skupín, aby bolo možné riešiť klasifikačnú úlohu. Keďže cieľový atribút závisí od recovered a confirmed, tak dané atribúty by sme pri trénovaní nemali použiť.

¹ <http://www.fao.org/faostat/en/#home>

² <https://www.prb.org/>

³ <https://coronavirus.jhu.edu/map.html>

Primárne budem pracovať so súborom Food_Supply_Quantity_kg_Data.csv, kde budú jednotlivé kroky popísané podrobnejšie.

V datasete niektoré informácie chýbajú. Riadky, kde chýbajú hodnoty. V riadku typu ako riadok 4 je možné dáta doplniť, NaN nahrádzam strednou hodnotou. V riadku typu ako riadok 80, kde chýbajú všetky dáta o COVID-19 je vhodné len odstrániť. V riadku typu ako riadok 29 sa hodnota dá dopočítať.

id	Country	Obesity	Undernourished	Confirmed	Deaths	Recovered	Active
4	Antigua and Barbuda	19.1	NaN	0.293878	0.007143	0.190816	0.095918
10	Bahamas	32.1	NaN	2.100763	0.044784	1.735115	0.320865
26	Canada	31.3	<2.5	2.109961	0.054203	1.909848	NaN
29	Chile	28.8	2.7	3.842229	0.097047	3.626194	NaN
52	French Polynesia	NaN	4.2	NaN	NaN	NaN	NaN
59	Grenada	20.2	NaN	0.130973	0.000885	0.129204	0.000885
80	Kiribati	45.6	2.7	NaN	NaN	NaN	NaN

Tab. 2 Ukážka chýbajúcich dát

Deskriptívna štatistika nám poskytne náhľad do charakteristiky hodnôt dát bez toho, aby sme ich museli po jednom študovať. Metódou describe() zistíme pre jednotlivé atribúty tieto charakteristiky: count (počet záznamov), mean (priemerná hodnota všetkých hodnôt tohto atribútu), std (štandardná odchýlka), min (minimálna hodnota), 25% (25. percentil hodnoty), 50%, 75%, max (maximálna hodnota).

	Alcoholic Beverages	Animal fats	Animal Products	Aquatic Products, Other	Cereals - Excluding Beer	Eggs	Fish, Seafood	...
count	164	164	164	164	164	164	164	...
mean	3.062504	0.225035	12.21286	0.013885	11.7808	0.47014	1.334066	...
std	2.383244	0.281732	5.902524	0.131623	5.873225	0.335623	1.190893	...
min	0	0.0018	1.7391	0	3.4014	0.0239	0.0342	...
25%	0.94355	0.040275	7.153025	0	7.047225	0.183875	0.547975	...
50%	2.893	0.118	12.09755	0	10.14275	0.45615	1.01945	...
75%	4.72465	0.254625	16.58458	0.001175	15.13105	0.64555	1.793975	...
max	15.3706	1.3559	26.8865	1.6794	29.8045	1.696	8.7959	...

Tab. 3 Ukážka deskriptívnej štatistiky

Vizualizácia dát

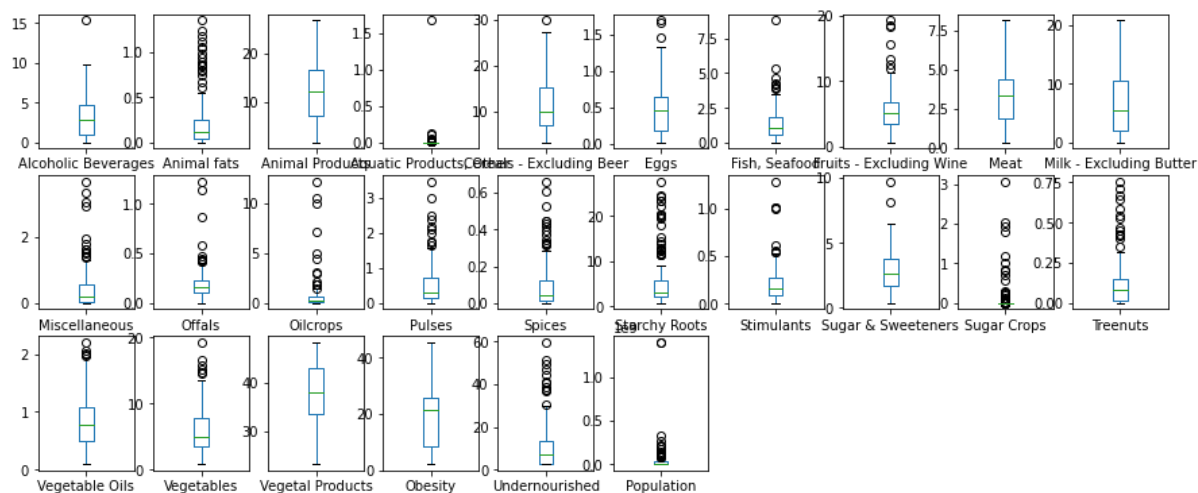
Ďalším popisným nástrojom údajov môže byť skreslenie, šikmosť alebo skosenie číselných atribútov. Pre dáta s normálnou distribúciou by hodnota skosenia mala byť blízka 0. Výsledky pre môj dataset ukazujú väčšie skreslenie atribútov Offals, Active, Oilcrops, Sugar Crops, Population, “Aquatic Products, Other“.

Vegetal Products	-0.27612
Obesity	-0.13496
Animal Products	0.276314
Meat	0.441191
Milk - Excluding Butter	0.562459
Vegetable Oils	0.801571
Cereals - Excluding Beer	0.886904
Eggs	1.002727
Sugar & Sweeteners	1.032276
Alcoholic Beverages	1.078912
Vegetables	1.138216
Deaths	1.173844
Confirmed	1.300358
Fruits - Excluding Wine	1.610589
Recovered	1.733656

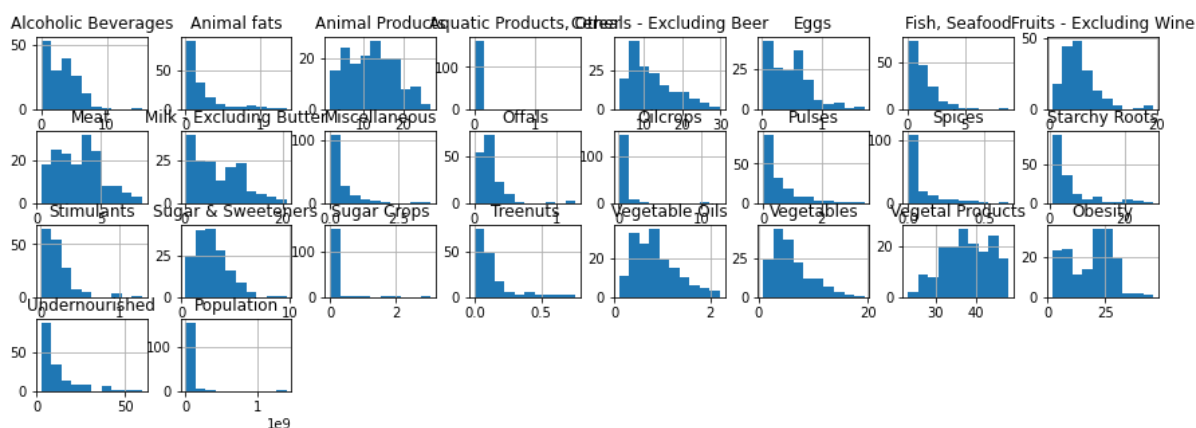
Undernourished	1.891053
Animal fats	2.071791
Pulses	2.083786
Starchy Roots	2.098638
Spices	2.250222
Treenuts	2.285705
Fish, Seafood	2.321963
Stimulants	2.583639
Miscellaneous	2.890873
Offals	3.430754
Active	3.713116
Oilcrops	5.122027
Sugar Crops	5.236852
Population	7.753474
Aquatic Products, Other	12.58844

Tab. 4 Skosenie jednotlivých atribútov

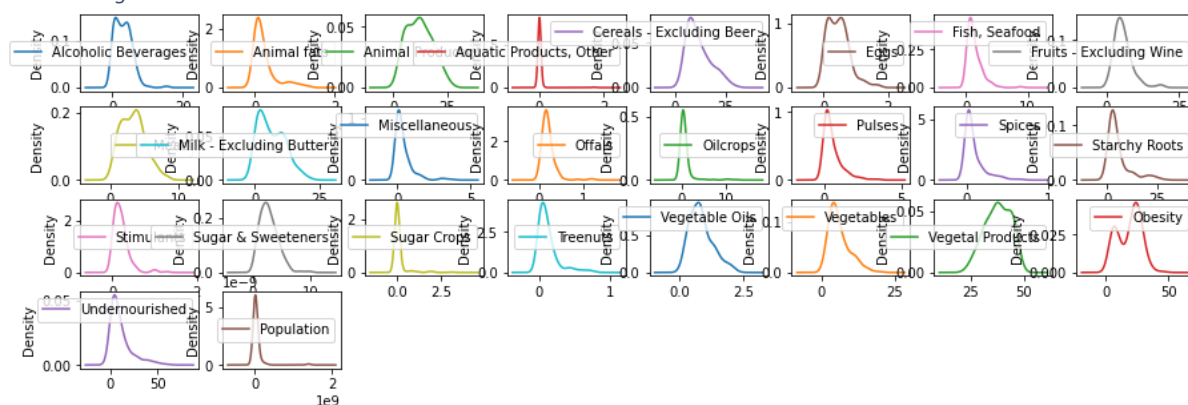
Aby sme lepšie porozumeli našim údajom, je užitočné si ich vizualizovať pomocou nasledujúcich metód. Skontrolujeme distribúciu každého atribútu pomocou histogramu, grafu hustoty a krabicových grafov.



Obr. 1 Krabicový graf pre atribúty



Obr. 2 Histogram atribútov



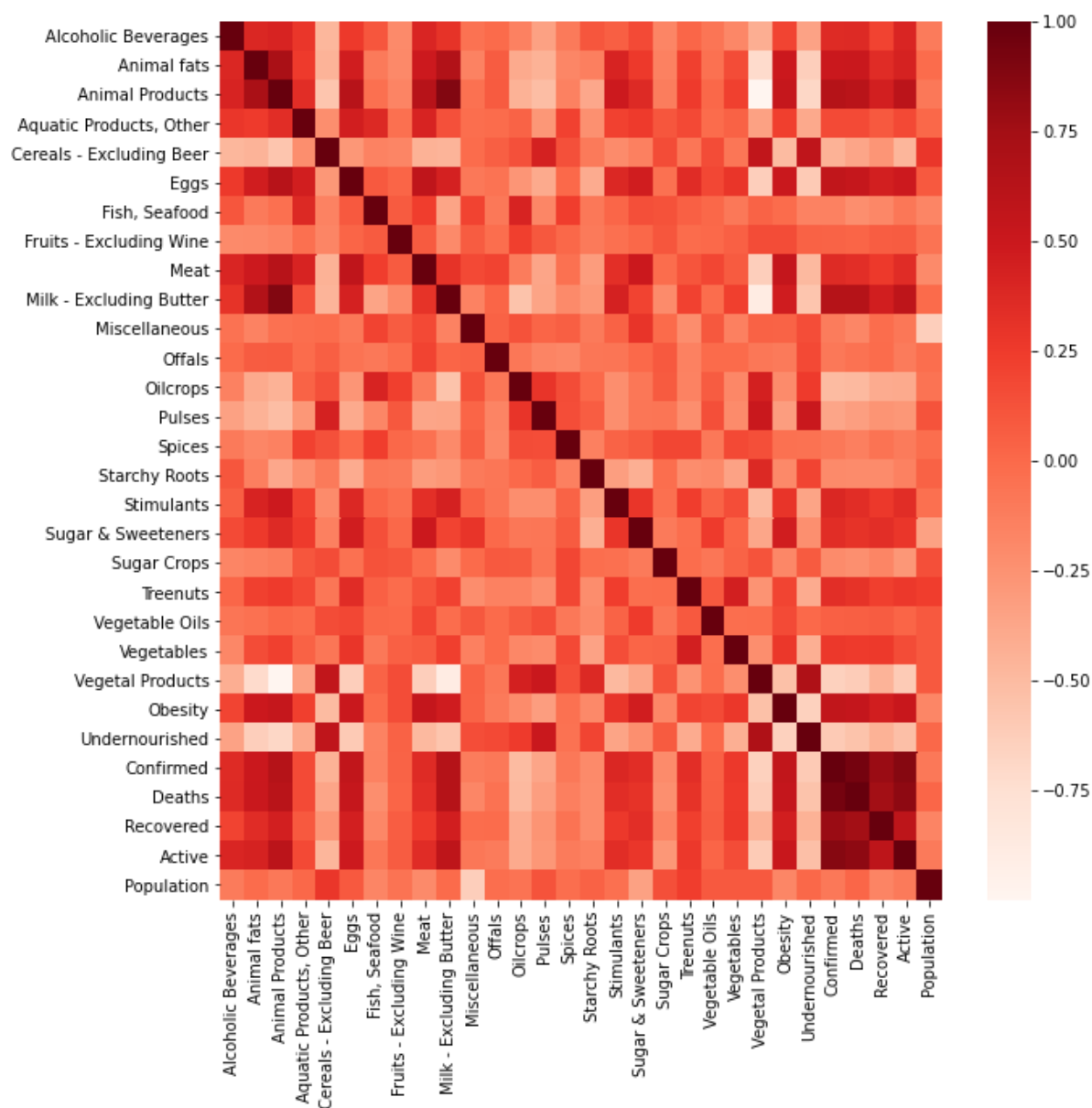
Obr. 3 Hustota atribútov

Z grafu hustoty atribútov a histogramu je vidieť, že nie všetky stĺpce majú normálne rozdelenie. (podobne ako pri skosení). Krabicový graf (Boxplot) popisuje jeden zo spôsobov grafickej vizualizácie numerických dát pomocou ich kvartilov. Stredná "krabicová" časť diagramu je zhora ohraničená 3. kvartilom, zdola 1. kvartilom a medzi nimi sa nachádza línia vymedzujúce medián. Boxploty môžu obsahovať tiež línie vychádzajúce zo strednej časti diagramu kolmo nahor a nadol, vyjadrujúce variabilitu dát pod prvým a nad tretím kvartilom. Odľahlé hodnoty (Outlier) sú vykreslené ako jednotlivé body.

Eliminácia črt:

Keďže nie všetky atribúty majú normálne rozdelenie je vhodné použiť `corr(method='spearman')`. Metódu 'pearson' je vhodnejšie použiť na dátach s normálnym rozdelením.

Ďalej som sledoval korelácie (vzťahy) medzi atribútmi. Z korelačných matíc na Obr. 4, Obr. je vidieť, že atribúty **Confirmed**, **Deaths**, **Recovered**, **Active** silne jeden od druhého závisia. Keďže umelo vytvorená cieľová premenná target je vytvorená z atribútov **Recovered** a **Confirmed**, tak tieto atribúty je potrebné odstrániť. Rovnako je vhodné odstrániť aj atribúty **Deaths**, **Active**, lebo veľmi úzko súvisia s atribútmi z ktorých sa dá priamo odvodiť cieľový atribút. Odstránené boli tiež stĺpce **Unit (all except Population)**, lebo obsahuje iba jeden znak '%' a **Country**, lebo obsahuje iba zoznam všetkých krajín.



Obr. 4 Korelačná matica na zobrazenie vzťahov medzi atribútmi



Obr. 5 Detail korelačnej matice

Škálovanie

V strojovom učení môže škálovanie funkcií zlepšiť rýchlosť konvergenzie rôznych algoritmov. Spočíva v úprave hodnôt do stanoveného menšieho rozsahu, obvykle do intervalu $[0,1]$ alebo $[-1,1]$, záleží od zvolenej transformácie. V tomto projekte som sa rozhodol pre preškálovanie na $[0,1]$ (min-max normalizácia).

Použité algoritmy

Rozhodovacie stromy:

Algoritmus rozhodovacích stromov môžeme považovať za učenie s dozorom vhodné na **klasifikáciu** a regresiu. Na začiatku je koreňovému uzlu priradený atribút a na základe hodnôt tohto atribútu sú tréningové dáta rozdelené na podmnožiny. Pre každú z týchto podmnožín je vytvorený uzol, ktorému je následne priradený ďalší atribút a takéto delenie pokračuje rekurzívne až kým nie je splnená ukončovacia podmienka. Z uzlov, v ktorých bola dosiahnutá ukončovacia podmienka sa stávajú listy a je im priradená jedna z možných tried. Možnosť pre výber najvhodnejšieho atribútu je viacero:

- **entropia** - miera homogenity v skupine príkladov (1 – rovnaký počet pozitívnych a negatívnych príkladov, 0 – len pozitívne príklady)
- **informačný zisk** - rozdiel entropie cieľového atribútu a celkovej entropie vysvetľujúceho atribútu v uzle
- **pomerový informačný zisk** - podiel informačného zisku a pomerovej entropie vysvetľujúceho atribútu v uzle
- **gini index** - pre kategorické atribúty, pravdepodobnosť umocnená na druhú
- **chí-kvadrát** - hľadá štatistickú súvislosť medzi cieľovým atribútom a iným atribútom. Vyberiem ten atribút, ktorý najviac súvisí s cieľovým

Je možné použiť aj menej prísne kritérium na ukončenie rastu, napríklad ak 90% príkladov spadá do tej istej triedy. Keď je strom príliš košatý na to, aby dostatočne zovšeobecňoval učiace dáta, tak sa používa osekávanie stromu – nahradenie nežiadúceho podstromu listom. Rozlišujeme medzi osekávaním počas učenia (pre-pruning) a osekávaním stromu po ukončení učenia (post-pruning). Výhodou rozhodovacích stromov je že sú rýchle na konštrukciu a veľmi prehľadné. V praxi sa používajú aj rôzne optimalizačné techniky ako bagging alebo boosting.

KNN klasifikátor (K-Nearest Neighbours):

Úlohou **KNN klasifikátora** je na základe k najbližších bodov predikovať hodnotu cieľového bodu. Počet vzoriek môže byť používateľom definovaná konštanta (k-nearest neighbor learning) alebo sa môže líšiť v závislosti od lokálnej hustoty bodov (radius-based neighbor learning). Vzdialenosť môže byť vo všeobecnosti ľubovoľná metrika, pričom najbežnejšia voľba je euklidovská vzdialenosť. Táto metóda je známa ako negeneralizujúca metóda strojového učenia, pretože si jednoducho „pamätá“ všetky svoje tréningové údaje (možno transformované do štruktúry rýchleho indexovania, napr. Ball Tree alebo KD Tree).

Keďže je to neparametrická metóda, je často úspešná v klasifikačných situáciách, keď je hranica rozhodnutia veľmi nepravidelná. Za nevýhodu sa považuje dlhší čas klasifikácie, kvôli prehľadávaniu veľkej časti príkladového priestoru.

Vo všeobecnosti prebieha KNN v nasledujúcich krokoch:

1. Načítanie údajov
2. Inicializácia K na vybraný počet susedov
3. Pre každý príklad v údajoch:
 - a) Vypočíta vzdialenosť medzi príkladom, ktorý chceme klasifikovať a aktuálnym príkladom z údajov
 - b) Do zoradenej kolekcie pridá vzdialenosť a index z príkladu
4. Usporiada zoradený súbor vzdialeností a indexov vo vzostupnom poradí podľa vzdialeností
5. Vyberie prvé K záznamy z triedenej kolekcie
6. Získa ohodnotenia vybraných K položiek
7. V prípade regresie vráti strednú hodnotu K ohodnotení, v prípade klasifikácie, vráti najviac zastúpenú triedu spomedzi K ohodnotení

Naivny Bayesov klasifikátor:

Skupina algoritmov s učením s učiteľom, ktoré sa snažia klasifikovať data na základe nejakého pravdepodobnostného modelu (nie nutne Bayesovského). To čo majú všetky takéto algoritmy spoločné je fakt, že jednotlivé atribúty považujú za vzájomne nezávislé. Samozrejme, vo väčšine prípadov to tak nie je, no tieto algoritmy sa považujú do istej miery za efektívne. V bayesovskej pravdepodobnosti je pravdepodobnosť udalosti A za predpokladu atribútov $x=(x_1, \dots, x_n)$ rovná (približne) pravdepodobnosti udalosti A krát súčin cez všetky pravdepodobnosti atribútov x iterujúce cez x_i za predpokladu udalosti A.

Tento algoritmus občas nie je najpresnejší no osvedčil sa pri klasifikácii spamu. Má niekoľko výhod: je rýchly, keďže pracuje s každým atribútom v podstate osobitne, veľké rozmery dát mu až tak nevadia.

Support vector machines (SVMs):

Algoritmy SVM môžeme považovať za učenie s dozorom vhodné na **klasifikáciu** a regresiu a detekciu outlier-ov. Princípom SVM je rozdelenie tréningových dát zakreslených v bodovom diagrame (Scatter plot) na dve protiľahlé oblasti náležiacie jednotlivým triedam dát. SVM priestor rozdeľuje nadrovinou (Hyperplane). Nelineárna SVM sa používa, keď rovinu dát nie je možné rozdeliť lineárne.

Výhody:

- Účinné vo vysokorozmerných priestoroch.
- Stále účinné v prípadoch, keď je počet dimenzií (atribútov) väčší ako počet vzoriek.
- Používa podmnožinu tréningových bodov vo funkcii rozhodovania (tzv. podporné vektory), takže je tiež pamäťovo efektívny.

Nevýhody:

- Ak je počet atribútov oveľa väčší ako počet vzoriek je potrebné veľmi opatrne používať predpripravené kernel a regularizáciu.
- Neposkytujú priamo odhady pravdepodobnosti, vypočítavajú sa pomocou nákladnej päťnásobnej krížovej validácie.

Rozdelenie dát na tréningové a testovacie

Máme nasledujúce možnosti:

- **K-zložková krížová validácia (K-fold cross-validation):** K-zložková krížová validácia funguje tak, že sa dáta rozdelia na K častí, potom sa vykoná tréningovanie K-krát, pričom počas 1 tréningu je (K-1) častí použitých na tréningovanie a tá zvyšná na testovanie. Takto sa to urobí K-krát, kým sa každá časť raz nestane tréningovou množinou. Tým získame K hodnôt presnosti, ktoré môžeme sumarizovať výpočtom priemernej hodnoty (mean). Vďaka preskúmaniu rôznych rozdelení dát získame presnosť s menšou variáciou.
- **Rozdeliť dataset** tak, že do istého indexu budú tréningové dáta a potom testovacie (väčšinou pomer tréningových a testovacích dát sú 2/3 k 1/3)

Evaluačné kritériá

Na interpretáciu kategorických výsledkov použijem nasledovné hodnoty:

Confusion matica pre binárne hodnoty:

- **TP (True Positives):** Hodnoty ktoré sú v skutočnosti pravdivé boli aj predikované ako pravdivé. Čím väčšia hodnota tým lepšie.
- **FP (False Positives):** Hodnoty, ktoré sú v skutočnosti nepravdivé, ale boli predikované ako pravdivé – Falošný poplach (Chyba Typu 1). Čím menšia hodnota tým lepšie.
- **FN (False Negatives):** Hodnoty, ktoré sú v skutočnosti pravdivé, ale boli predikované ako nepravdivé – Minutie (Chyba Typu 2). Čím menšia hodnota tým lepšie.
- **TN (True Negatives):** Hodnoty, ktoré sú v skutočnosti nepravdivé boli aj predikované ako nepravdivé. Čím väčšia hodnota tým lepšie.

Balanced Accuracy: $(TP/(TP+FN) + TN/(TN+FP)) / 2 = (\text{Sensitivity} + \text{Specificity}) / 2$

Precision: $TP/(TP+FP)$. Je to schopnosť klasifikátora neoznačiť vzorku, ktorá je v skutočnosti negatívna za pozitívnu. Čím bližšie k 1, tým lepšie.

Recall: $TP/(TP+FN)$. Schopnosť klasifikátora nájsť všetky pozitívne prípady. Čím bližšie k 1, tým lepšie.

F1 skóre: harmonický priemer precision-u a recall-u: Zahrňuje precision aj recall, ale dáva im rovnakú váhu čo nie je veľmi dobré v prípade nevyváženého cieľového atribútu alebo v prípade keď jeden typ chyby je preferovaný pred druhou. Čím väčšie skóre tým lepšie. najlepšie skóre je 1.0 a najhoršie 0.0.

Support: počet skutočných výskytov triedy v špecifikovanom súbore údajov.

ROC oblasť(receiver operating characteristic): ROC krivka ukazuje vzťah pomeru TP a FP. (Do grafu kde x-ová os reprezentuje FP-rate a y-ová TP-rate, zakreslí krivku pozostávajúcu z bodov predikcie). Dáva nám informáciu o tom, že pri nejakom FP/TP budeme mať aké TP/FP. Dobré na nej je, že vyjadruje rovnako presný klasifikátor, ale znázorňuje náklonnosť k jednému typu odpovede (pre prípady, že jednu chybu máme v úmysle zredukovať viac hoci aj na úkor druhej). Oblasťou sa rozumie oblasť pod krivkou.

Popis výsledkov

Kvôli veľkej nevyváženosti datasetu (v prípade ak sa na cieľovom atribúte spravila min-max normalizácia na interval [0,1], a čísla menšie ako 0.5 patrili do skupiny 0 a čísla väčšie ako 0.5 patrili do skupiny 1. Pomer skupín 0 a 1 bol 18:146. Keď sa použila K-fold krížová validácia, tak to znamenalo, že v jednom folde boli iba 2 negatívne a 14 pozitívnych

príkladov, čo ale spôsobilo pri testovaných algoritmoch, že pre skupinu 0 boli všetky atribúty z klasifikačného reportu rovné 0). Preto bolo potrebné zmeniť hranicu pre rozdelenie skupín na 0.75 a použiť **over-sampling**, čím sa dosiahol rovnaký počet príkladov v minoritnej a majoritnej skupine. Tento postup bráni modelu k prikloneniu sa k väčšinovej triede.

Keďže chceme testovať tento dataset aj v prípade, že cieľový atribút nebude binárny je potrebné nájsť také upsampling implementácie, ktoré zvládajú aj **multiclass klasifikáciu** a nie len binárnu klasifikáciu. Rozhodol som sa pre nasledujúce implementácie:

MDO (Mahalanobis Distance-based Over-sampling technique): generuje syntetické vzorky, ktoré majú rovnakú Mahalanobis-ovu vzdialenosť od strednej hodnoty uvažovanej triedy ako ostatné minoritné triedy. Zachovaním kovariančnej štruktúry inštancií minoritných tried a inteligentným generovaním syntetických vzoriek pozdĺž obrysov pravdepodobnosti sú nové inštanície minoritných tried lepšie modelované pre algoritmy strojového učenia.

SMOTE (Synthetic Minority Oversampling Technique) alebo ADASYN: používajú na generovanie nových vzoriek rovnaký algoritmus. Ak vezmeme do úvahy vzorku x_i , vygeneruje sa nová vzorka x_{new} s ohľadom na jej k najbližších susedov. Potom je vybraný jeden z týchto najbližších susedov x_{zi} a vzorka je vygenerovaná takto: $x_{new} = x_i + \lambda \cdot (x_{zi} - x_i)$, kde λ je náhodné číslo z intervalu $[0,1]$. Táto interpolácia vytvorí vzorku na priamke medzi x_i a x_{zi} , ADASYN sa od SMOTE líši tým, že počet vzoriek vygenerovaný pomocou x_i je proporčný počtu vzoriek, ktoré nie sú z tej istej triedy ako x_i v danom susedstve.

Poznámka: Klasifikačný report je vygenerovaný pomocou funkcie `cross_val_predict`, ktorá funguje tak, že natrénuje model na $k-1$ foldoch a na k -tom folde testuje, toto opakuje k -krát a postupne pridáva výsledky do záverečného klasifikačného reportu a aj confusion matice.

Výsledky vybraných algoritmů

Target atribút: **2 skupiny (0 pre [0, 0.75), 1 pre [0.75, 1])**

Pomer tried pred oversamplingom: **121:43**

Oversampling algoritmus: **ADASYN**

Algoritmy: DecisionTreeClassifier, RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier, XGBClassifier, BaggingClassifier(XGBClassifier), KNeighborsClassifier, GaussianNB, SVC

DT_def {'fit_time': 0.0041, 'score_time': 0.0052, 'test_balanced_accuracy': 0.7365, 'test_precision_weighted': 0.7523, 'test_recall_weighted': 0.7375, 'test_f1_weighted': 0.7324, 'test_roc_auc_ovo_weighted': 0.7365}

	precision	recall	f1-score	support		
0	0.71	0.81	0.76	123	0	100
1	0.78	0.66	0.71	121	1	23
accuracy			0.74	244		41
macro avg	0.74	0.74	0.74	244		80
weighted avg	0.74	0.74	0.74	244	0	1

DT_ent {'fit_time': 0.0076, 'score_time': 0.0059, 'test_balanced_accuracy': 0.7449, 'test_precision_weighted': 0.752, 'test_recall_weighted': 0.7458, 'test_f1_weighted': 0.7444, 'test_roc_auc_ovo_weighted': 0.7449}

	precision	recall	f1-score	support		
0	0.74	0.77	0.75	123	0	95
1	0.76	0.72	0.74	121	1	28
accuracy			0.75	244		34
macro avg	0.75	0.75	0.75	244		87
weighted avg	0.75	0.75	0.75	244	0	1

DT_depth {'fit_time': 0.0035, 'score_time': 0.0068, 'test_balanced_accuracy': 0.6824, 'test_precision_weighted': 0.7046, 'test_recall_weighted': 0.6847, 'test_f1_weighted': 0.6742, 'test_roc_auc_ovo_weighted': 0.7669}

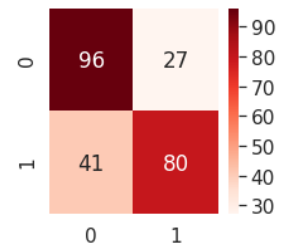
	precision	recall	f1-score	support		
0	0.65	0.83	0.73	123	0	102
1	0.76	0.54	0.63	121	1	21
accuracy			0.68	244		56
macro avg	0.70	0.68	0.68	244		65
weighted avg	0.70	0.68	0.68	244	0	1

DT_split {'fit_time': 0.0045, 'score_time': 0.0065, 'test_balanced_accuracy': 0.7532, 'test_precision_weighted': 0.7689, 'test_recall_weighted': 0.754, 'test_f1_weighted': 0.7497, 'test_roc_auc_ovo_weighted': 0.7605}

	precision	recall	f1-score	support		
0	0.73	0.81	0.77	123	0	100
1	0.79	0.69	0.74	121	1	23
accuracy			0.75	244		37
macro avg	0.76	0.75	0.75	244		84
weighted avg	0.76	0.75	0.75	244	0	1

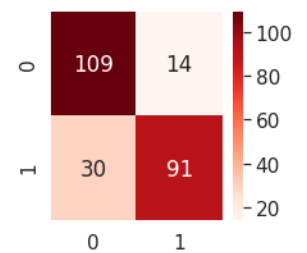
DT_leaf {'fit_time': 0.0039, 'score_time': 0.0066,
'test_balanced_accuracy': 0.7215, 'test_precision_weighted': 0.7311,
'test_recall_weighted': 0.7208, 'test_f1_weighted': 0.717,
'test_roc_auc_ovo_weighted': 0.7446}

	precision	recall	f1-score	support
0	0.70	0.78	0.74	123
1	0.75	0.66	0.70	121
accuracy			0.72	244
macro avg	0.72	0.72	0.72	244
weighted avg	0.72	0.72	0.72	244



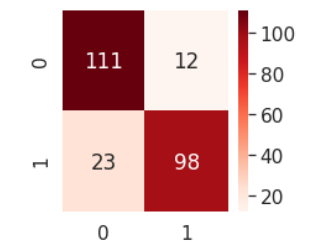
RF1 {'fit_time': 0.916, 'score_time': 0.1514, 'test_balanced_accuracy':
0.8183, 'test_precision_weighted': 0.8309, 'test_recall_weighted': 0.8193,
'test_f1_weighted': 0.8172, 'test_roc_auc_ovo_weighted': 0.8822}

	precision	recall	f1-score	support
0	0.78	0.89	0.83	123
1	0.87	0.75	0.81	121
accuracy			0.82	244
macro avg	0.83	0.82	0.82	244
weighted avg	0.83	0.82	0.82	244



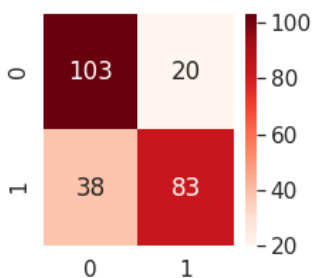
RF2 {'fit_time': 0.9699, 'score_time': 0.1575, 'test_balanced_accuracy':
0.8558, 'test_precision_weighted': 0.8623, 'test_recall_weighted': 0.8565,
'test_f1_weighted': 0.8559, 'test_roc_auc_ovo_weighted': 0.927}

	precision	recall	f1-score	support
0	0.83	0.90	0.86	123
1	0.89	0.81	0.85	121
accuracy			0.86	244
macro avg	0.86	0.86	0.86	244
weighted avg	0.86	0.86	0.86	244



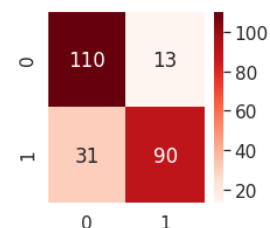
AB {'fit_time': 0.006, 'score_time': 0.0068, 'test_balanced_accuracy':
0.7615, 'test_precision_weighted': 0.7732, 'test_recall_weighted': 0.7618,
'test_f1_weighted': 0.7582, 'test_roc_auc_ovo_weighted': 0.7615}

	precision	recall	f1-score	support
0	0.73	0.84	0.78	123
1	0.81	0.69	0.74	121
accuracy			0.76	244
macro avg	0.77	0.76	0.76	244
weighted avg	0.77	0.76	0.76	244



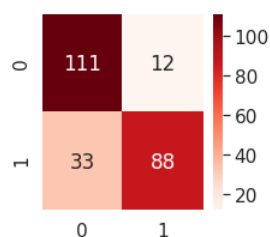
```
GB {'fit_time': 0.8979, 'score_time': 0.0071, 'test_balanced_accuracy':
0.8189, 'test_precision_weighted': 0.8332, 'test_recall_weighted': 0.8192,
'test_f1_weighted': 0.8144, 'test_roc_auc_ovo_weighted': 0.8986}
```

	precision	recall	f1-score	support
0	0.78	0.89	0.83	123
1	0.87	0.74	0.80	121
accuracy			0.82	244
macro avg	0.83	0.82	0.82	244
weighted avg	0.83	0.82	0.82	244



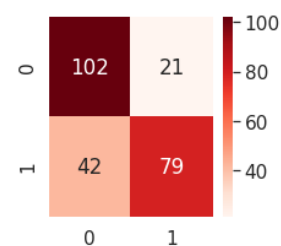
```
xgb {'fit_time': 0.1843, 'score_time': 0.0071, 'test_balanced_accuracy':
0.8154, 'test_precision_weighted': 0.8274, 'test_recall_weighted': 0.8158,
'test_f1_weighted': 0.8134, 'test_roc_auc_ovo_weighted': 0.8863}
```

	precision	recall	f1-score	support
0	0.77	0.90	0.83	123
1	0.88	0.73	0.80	121
accuracy			0.82	244
macro avg	0.83	0.81	0.81	244
weighted avg	0.82	0.82	0.81	244



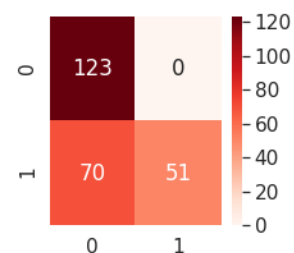
```
Bag_xgb {'fit_time': 0.9603, 'score_time': 0.0164,
'test_balanced_accuracy': 0.7397, 'test_precision_weighted': 0.7526,
'test_recall_weighted': 0.7417, 'test_f1_weighted': 0.7375,
'test_roc_auc_ovo_weighted': 0.7939}
```

	precision	recall	f1-score	support
0	0.71	0.83	0.76	123
1	0.79	0.65	0.71	121
accuracy			0.74	244
macro avg	0.75	0.74	0.74	244
weighted avg	0.75	0.74	0.74	244



```
knn {'fit_time': 0.0007, 'score_time': 0.0083, 'test_balanced_accuracy':
0.7103, 'test_precision_weighted': 0.8195, 'test_recall_weighted': 0.7133,
'test_f1_weighted': 0.6805, 'test_roc_auc_ovo_weighted': 0.8243}
```

	precision	recall	f1-score	support
0	0.64	1.00	0.78	123
1	1.00	0.42	0.59	121
accuracy			0.71	244
macro avg	0.82	0.71	0.69	244
weighted avg	0.82	0.71	0.69	244



```

NB {'fit_time': 0.0014, 'score_time': 0.0077, 'test_balanced_accuracy':
0.6035, 'test_precision_weighted': 0.6807, 'test_recall_weighted': 0.6065,
'test_f1_weighted': 0.562, 'test_roc_auc_ovo_weighted': 0.6688}
precision    recall    f1-score   support

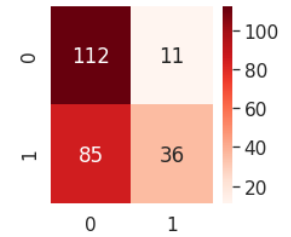
```

```

      0      0.57      0.91      0.70      123
      1      0.77      0.30      0.43      121

accuracy          0.61      244
macro avg          0.67      0.60      0.56      244
weighted avg       0.67      0.61      0.57      244

```



```

svm {'fit_time': 0.02, 'score_time': 0.0071, 'test_balanced_accuracy':
0.7436, 'test_precision_weighted': 0.7662, 'test_recall_weighted': 0.7455,
'test_f1_weighted': 0.7407, 'test_roc_auc_ovo_weighted': 0.8008}
precision    recall    f1-score   support

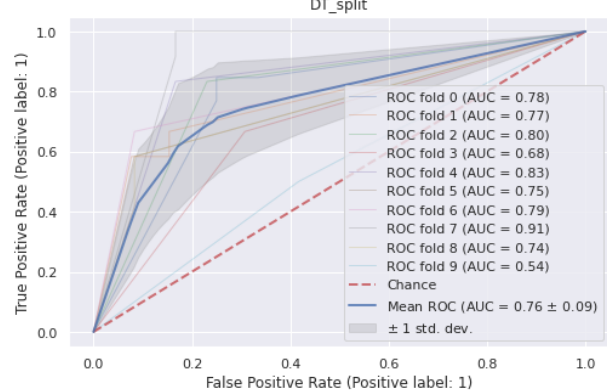
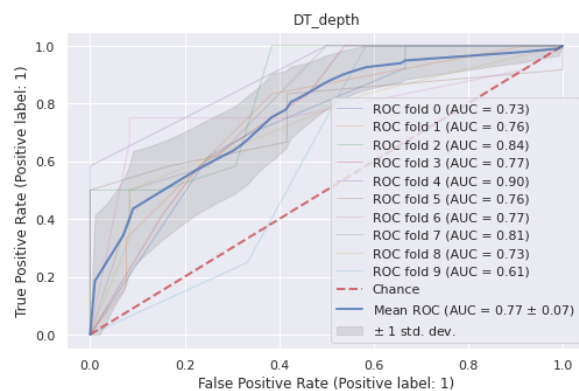
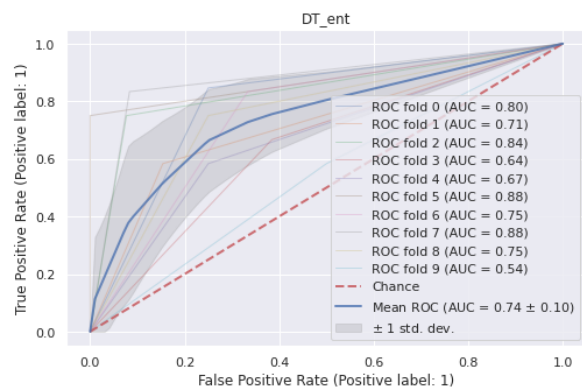
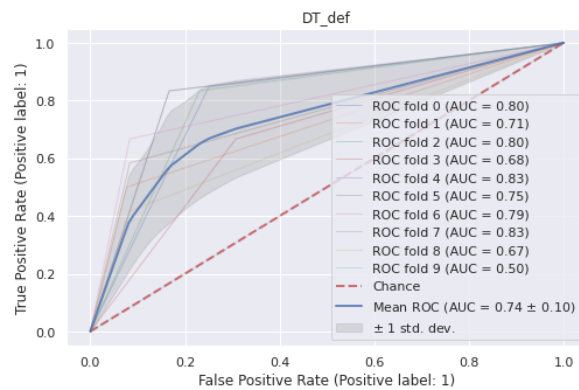
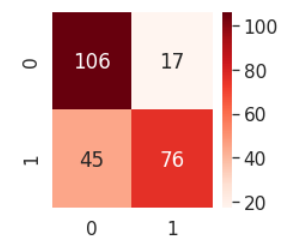
```

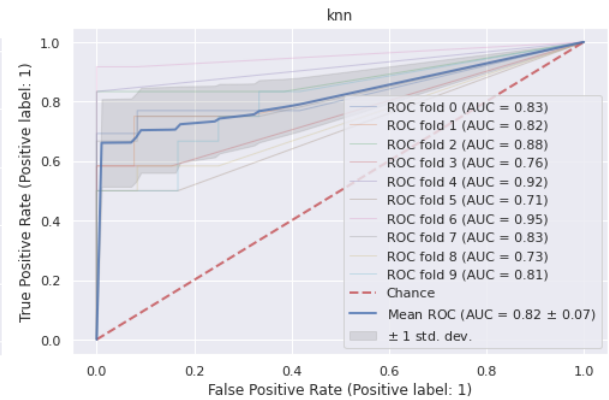
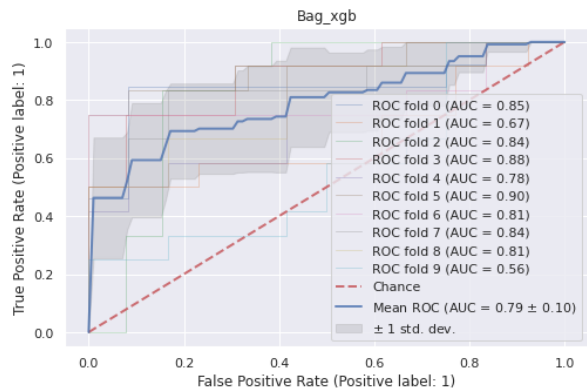
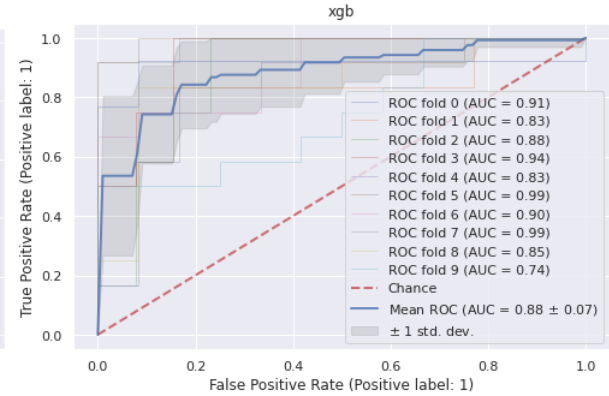
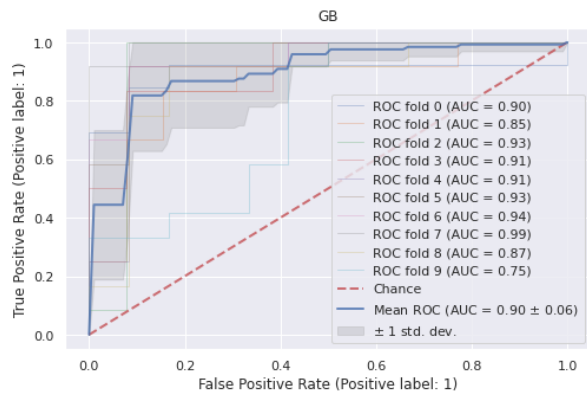
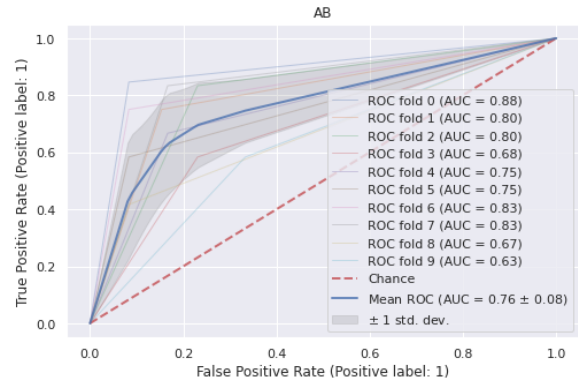
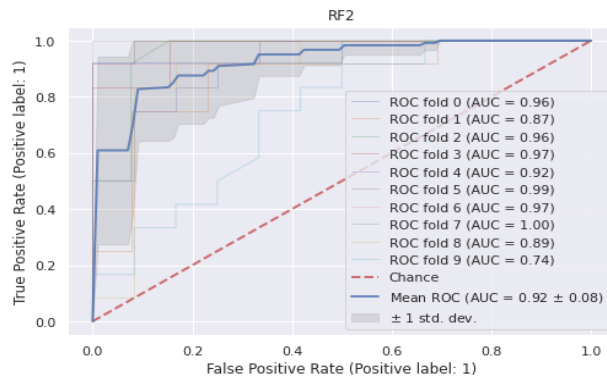
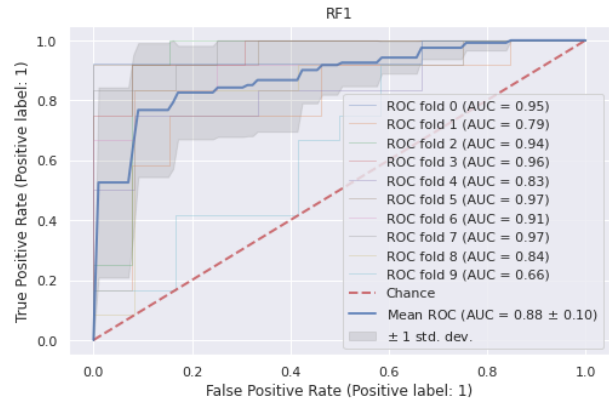
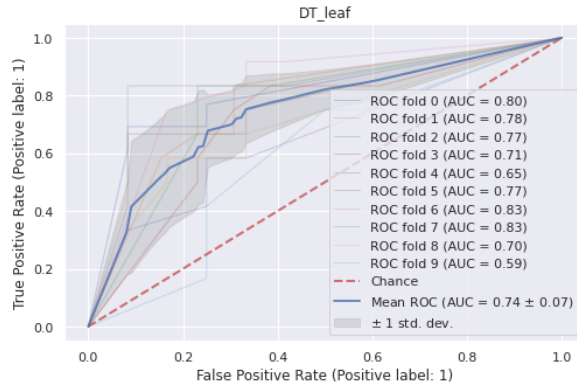
```

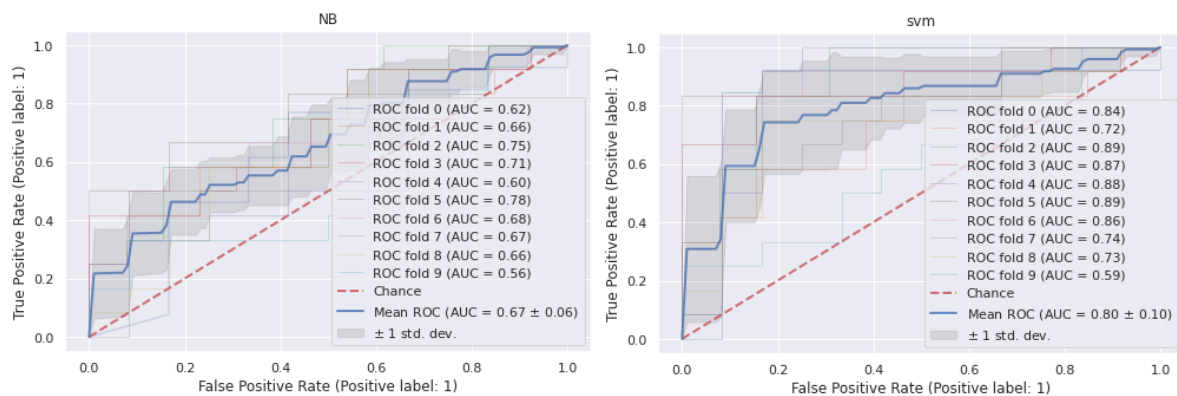
      0      0.70      0.86      0.77      123
      1      0.82      0.63      0.71      121

accuracy          0.75      244
macro avg          0.76      0.74      0.74      244
weighted avg       0.76      0.75      0.74      244

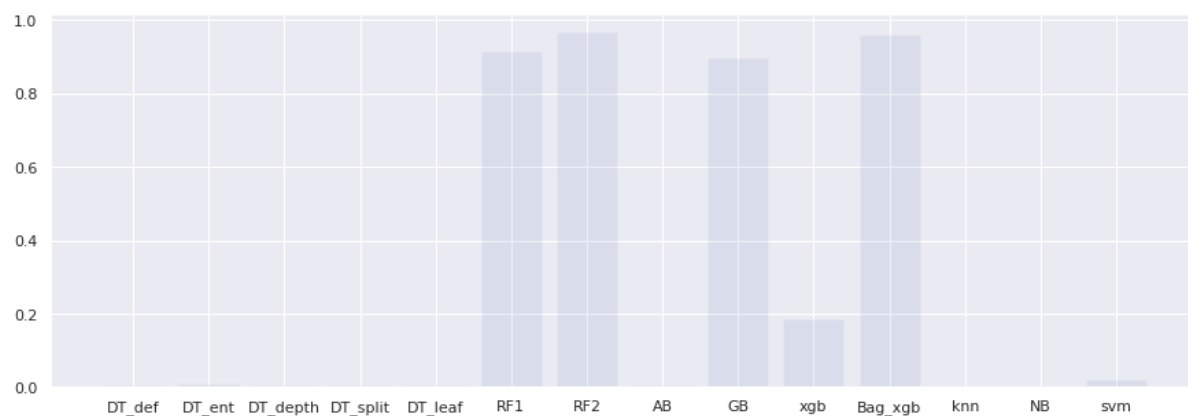
```



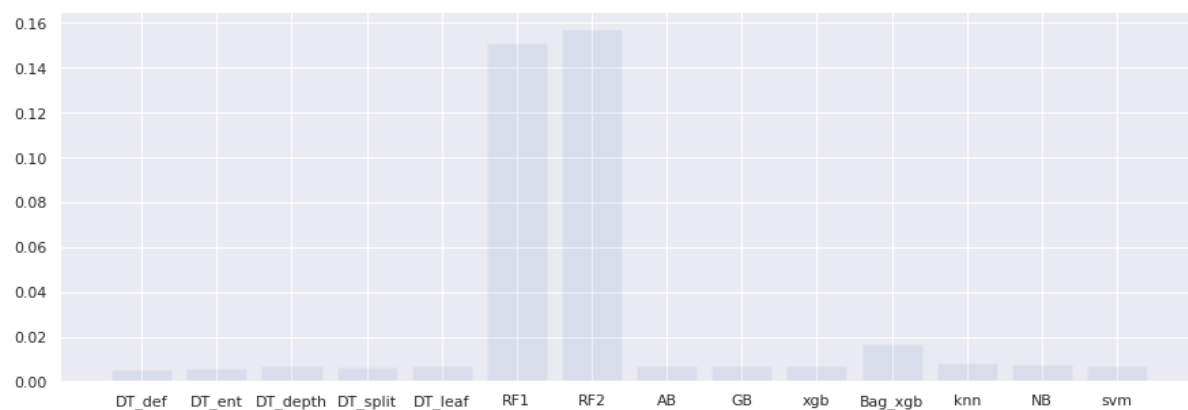




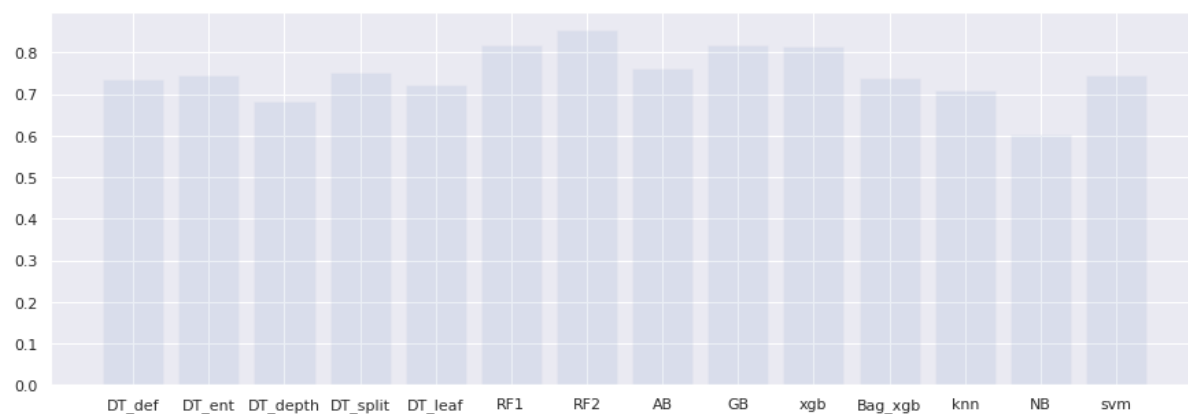
Comparison of fit_time feature of classification algorithms



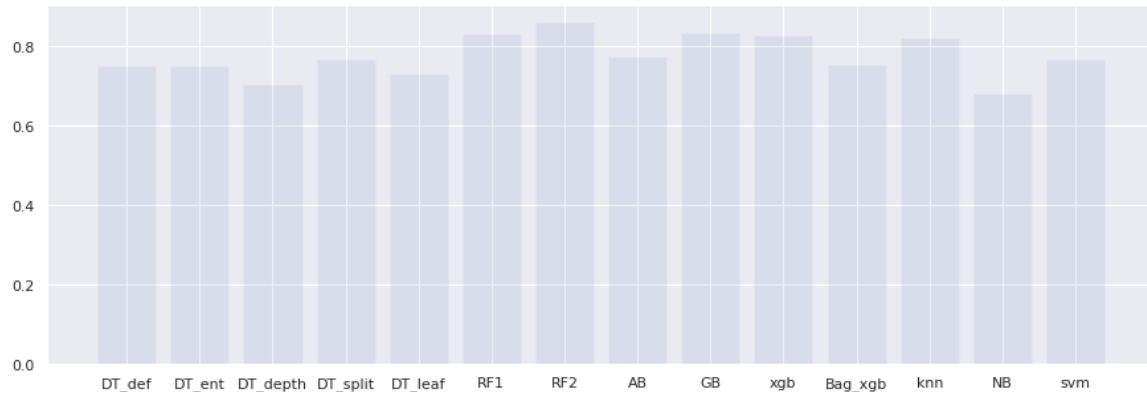
Comparison of score_time feature of classification algorithms



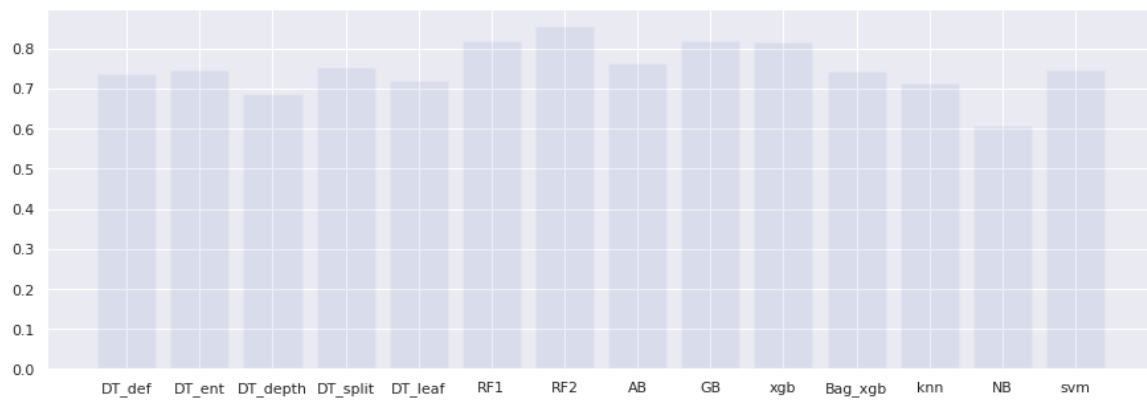
Comparison of test_balanced_accuracy feature of classification algorithms



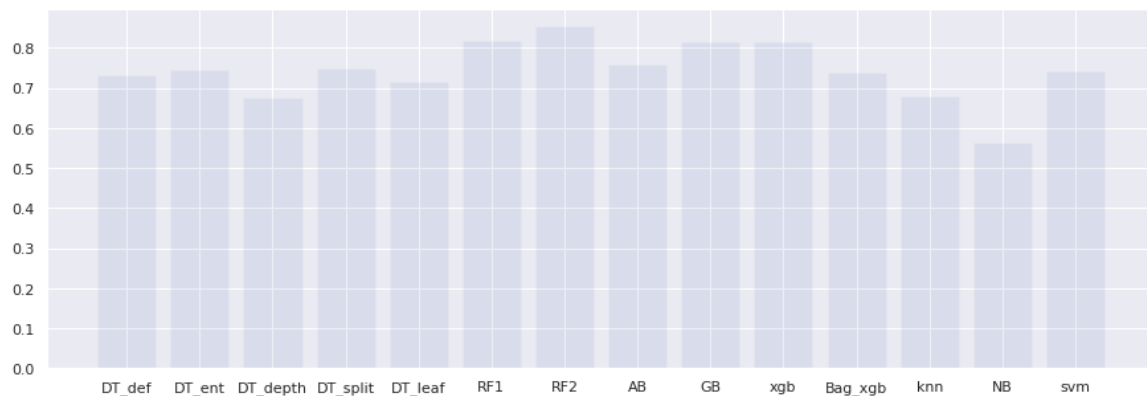
Comparison of test_precision_weighted feature of classification algorithms



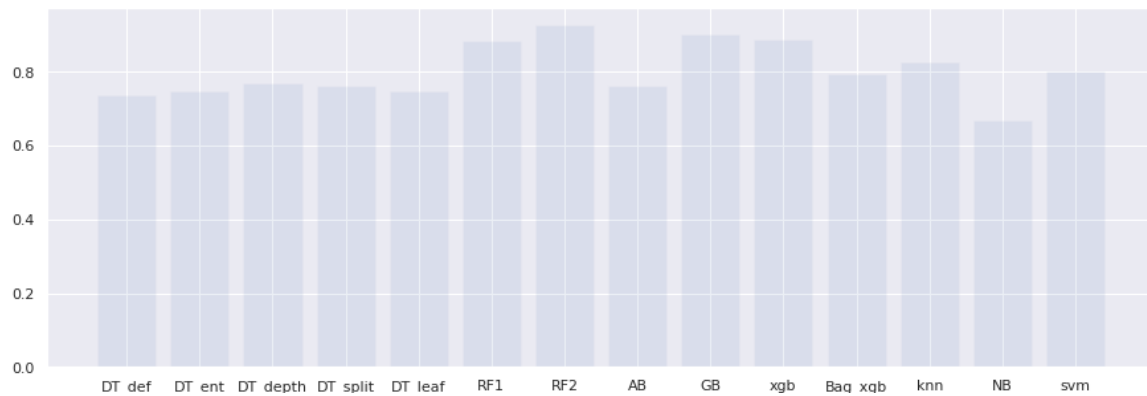
Comparison of test_recall_weighted feature of classification algorithms



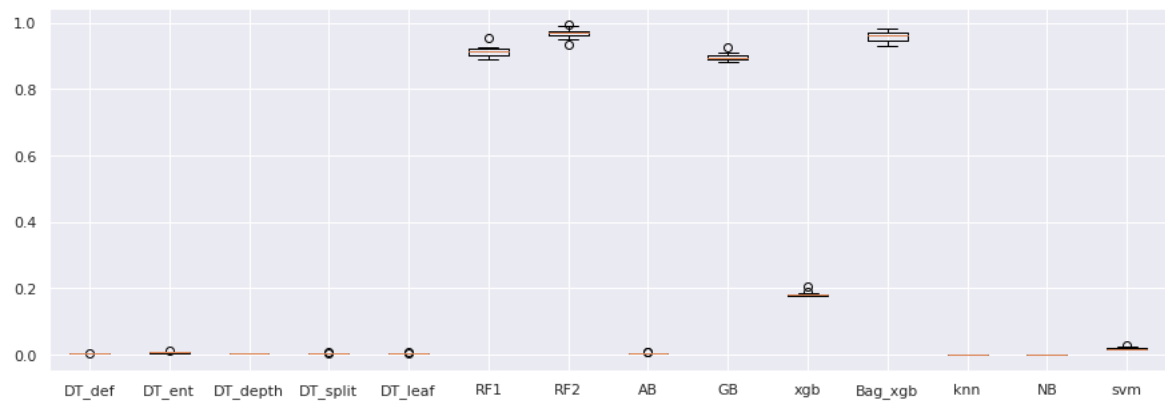
Comparison of test_f1_weighted feature of classification algorithms



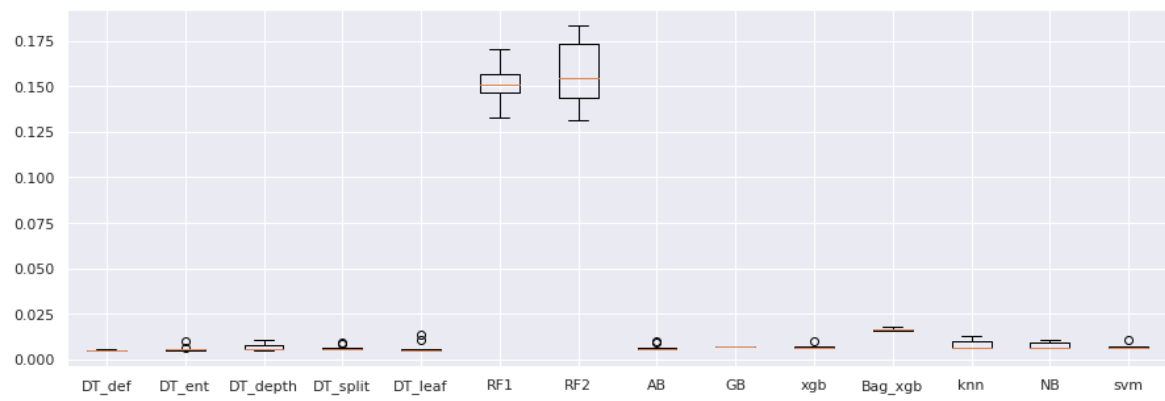
Comparison of test_roc_auc_ovo_weighted feature of classification algorithms



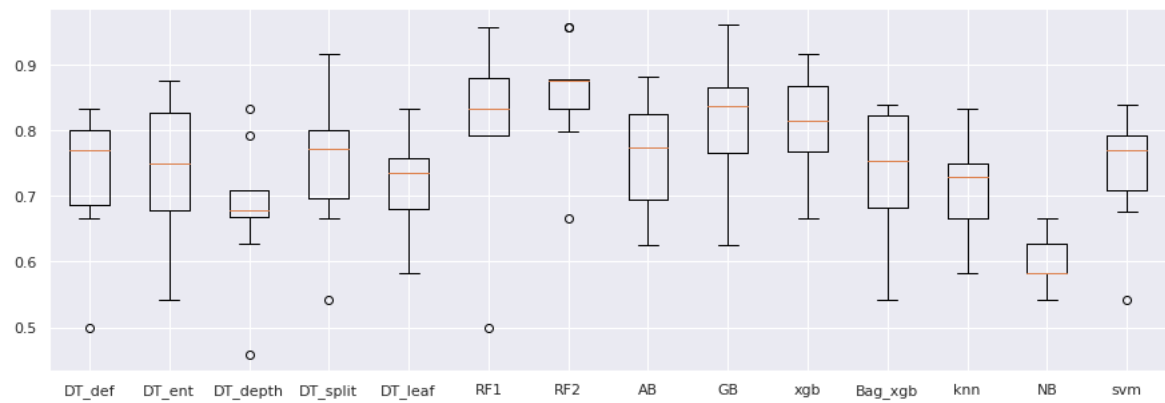
Comparison of score fit_time of classification algorithms



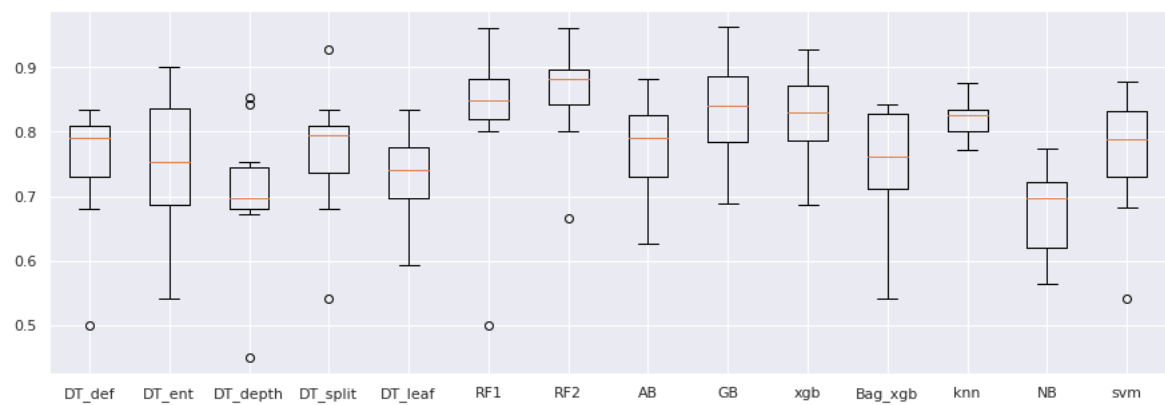
Comparison of score score_time of classification algorithms



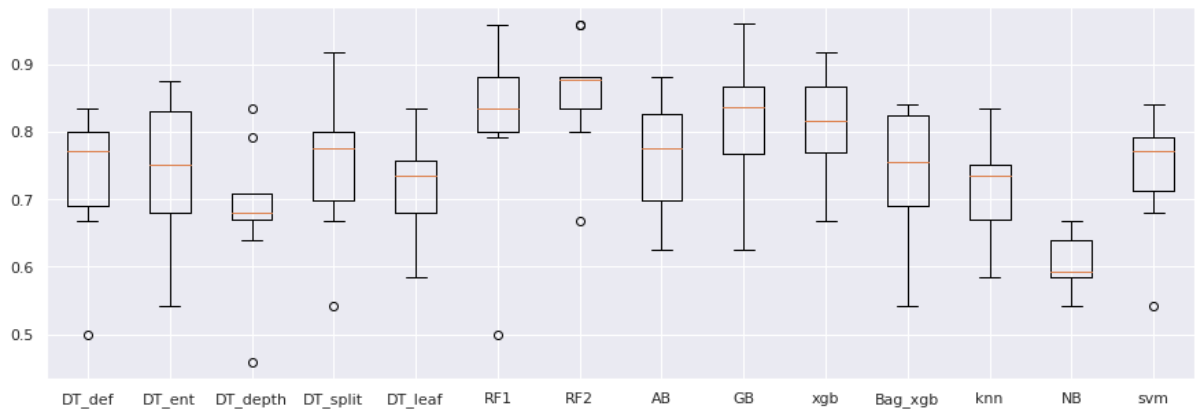
Comparison of score test_balanced_accuracy of classification algorithms



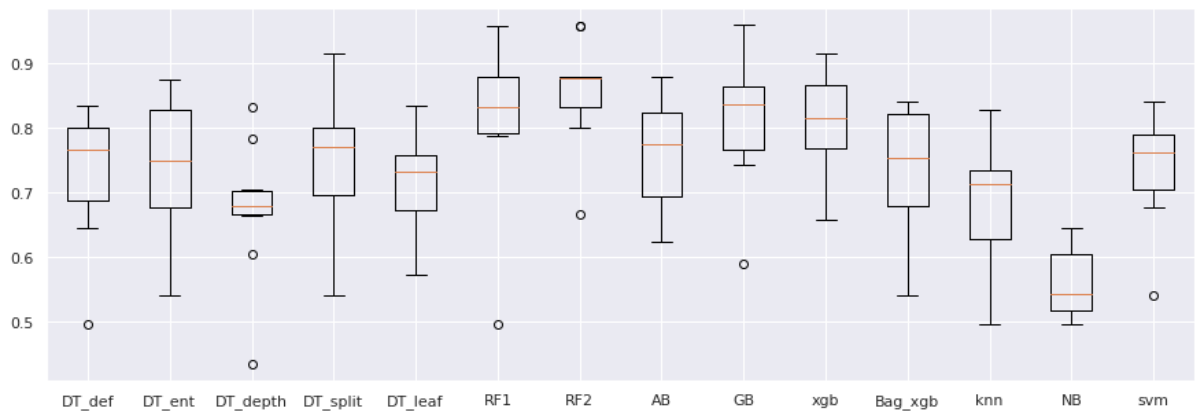
Comparison of score test_precision_weighted of classification algorithms



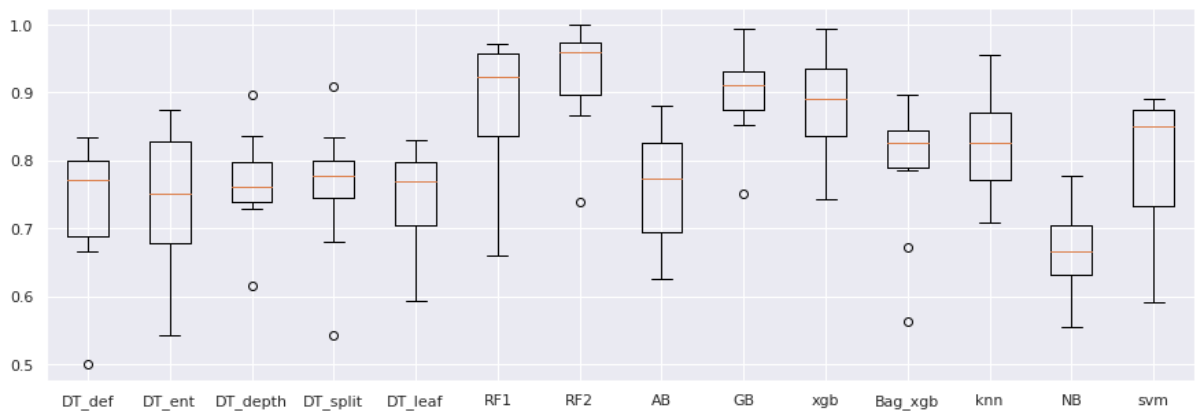
Comparison of score test_recall_weighted of classification algorithms

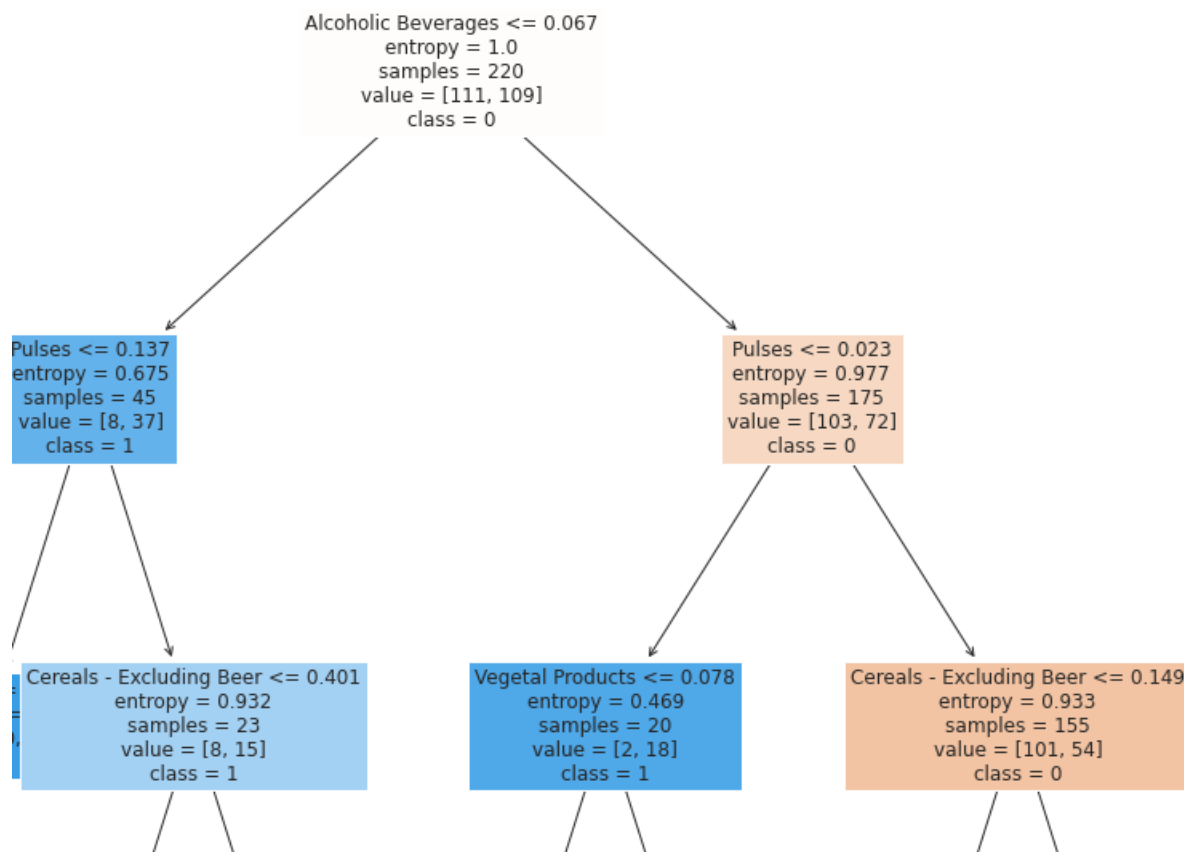


Comparison of score test_f1_weighted of classification algorithms



Comparison of score test_roc_auc_ovo_weighted of classification algorithms





Obr. 4 Výřez default decision tree

Z tohto reportu je ľahko vidieť, že najlepšie výsledky sa podarilo dosiahnuť pomocou metódy RandomForest, kde precision bol na úrovni 86%, avšak má najdlhší čas tréovania. Druhé najlepšie výsledky sa podarilo dosiahnuť pomocou GradientBoostingu a to precision na úrovni 83% a tretí najlepší výsledok sa podarilo získať pomocou ExtremeGradientBoostingu a to precision na úrovni 82%. Za nainovatívnejšiu metódu sa považuje práve ExtremeGradientBoosting, ktorý poskytol cca. 5 krát kratší čas tréovania ako RandomForest. Potvrdilo sa aj očakávanie, že najhoršie výsledky dosiahne práve NaiveBayes.

Výřez z rozhodovacieho stromu naznačuje, že najdôležitejšie atribúty na to, aby sa podarilo covid prekonať sú alkohol, strukoviny (pulses) a obilniny (cereals).

Target atribút: **2 skupiny (0 pre [0, 0.75), 1 pre [0.75, 1])**

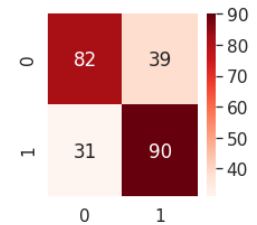
Pomer tried pred oversamplingom: **121:43**

Oversampling algoritmus: **MDO**

Algoritmy: DecisionTreeClassifier, RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier, XGBClassifier, BaggingClassifier(XGBClassifier), KNeighborsClassifier, GaussianNB, SVC

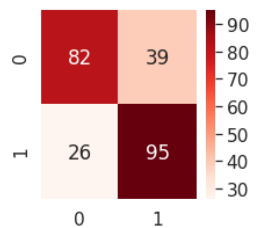
DT_def {'fit_time': 0.0051, 'score_time': 0.0061, 'test_balanced_accuracy': 0.7109, 'test_precision_weighted': 0.7211, 'test_recall_weighted': 0.7105, 'test_f1_weighted': 0.7072, 'test_roc_auc_ovo_weighted': 0.7109}

	precision	recall	f1-score	support
0	0.73	0.68	0.70	121
1	0.70	0.74	0.72	121
accuracy			0.71	242
macro avg	0.71	0.71	0.71	242
weighted avg	0.71	0.71	0.71	242



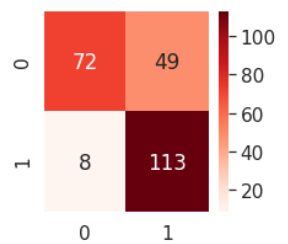
DT_ent {'fit_time': 0.0093, 'score_time': 0.0064, 'test_balanced_accuracy': 0.7324, 'test_precision_weighted': 0.7453, 'test_recall_weighted': 0.7323, 'test_f1_weighted': 0.7281, 'test_roc_auc_ovo_weighted': 0.7324}

	precision	recall	f1-score	support
0	0.76	0.68	0.72	121
1	0.71	0.79	0.75	121
accuracy			0.73	242
macro avg	0.73	0.73	0.73	242
weighted avg	0.73	0.73	0.73	242



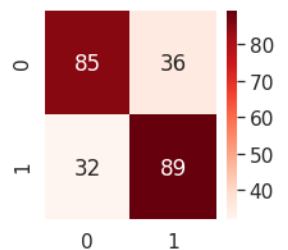
DT_depth {'fit_time': 0.0033, 'score_time': 0.0059, 'test_balanced_accuracy': 0.7651, 'test_precision_weighted': 0.8124, 'test_recall_weighted': 0.7648, 'test_f1_weighted': 0.7496, 'test_roc_auc_ovo_weighted': 0.7938}

	precision	recall	f1-score	support
0	0.90	0.60	0.72	121
1	0.70	0.93	0.80	121
accuracy			0.76	242
macro avg	0.80	0.76	0.76	242
weighted avg	0.80	0.76	0.76	242



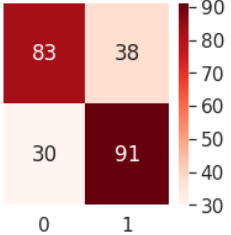
DT_split {'fit_time': 0.0049, 'score_time': 0.0062, 'test_balanced_accuracy': 0.7192, 'test_precision_weighted': 0.7274, 'test_recall_weighted': 0.7188, 'test_f1_weighted': 0.7165, 'test_roc_auc_ovo_weighted': 0.7265}

	precision	recall	f1-score	support
0	0.73	0.70	0.71	121
1	0.71	0.74	0.72	121
accuracy			0.72	242
macro avg	0.72	0.72	0.72	242
weighted avg	0.72	0.72	0.72	242



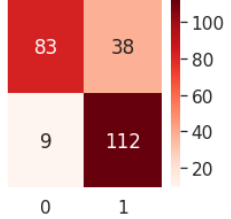
DT_leaf {'fit_time': 0.0046, 'score_time': 0.0063,
'test_balanced_accuracy': 0.7196, 'test_precision_weighted': 0.7266,
'test_recall_weighted': 0.7197, 'test_f1_weighted': 0.7158,
'test_roc_auc_ovo_weighted': 0.763}

	precision	recall	f1-score	support		
0	0.73	0.69	0.71	121	0	83
1	0.71	0.75	0.73	121	1	38
accuracy			0.72	242		
macro avg	0.72	0.72	0.72	242		
weighted avg	0.72	0.72	0.72	242		



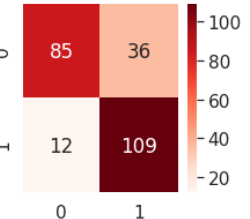
RF1 {'fit_time': 0.9209, 'score_time': 0.1534, 'test_balanced_accuracy':
0.8067, 'test_precision_weighted': 0.834, 'test_recall_weighted': 0.8067,
'test_f1_weighted': 0.8001, 'test_roc_auc_ovo_weighted': 0.8325}

	precision	recall	f1-score	support		
0	0.90	0.69	0.78	121	0	83
1	0.75	0.93	0.83	121	1	38
accuracy			0.81	242		
macro avg	0.82	0.81	0.80	242		
weighted avg	0.82	0.81	0.80	242		



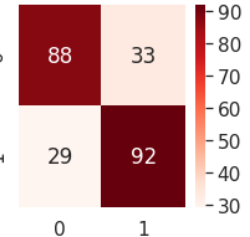
RF2 {'fit_time': 0.9921, 'score_time': 0.1587, 'test_balanced_accuracy':
0.8022, 'test_precision_weighted': 0.8236, 'test_recall_weighted': 0.8025,
'test_f1_weighted': 0.7973, 'test_roc_auc_ovo_weighted': 0.8197}

	precision	recall	f1-score	support		
0	0.88	0.70	0.78	121	0	85
1	0.75	0.90	0.82	121	1	36
accuracy			0.80	242		
macro avg	0.81	0.80	0.80	242		
weighted avg	0.81	0.80	0.80	242		



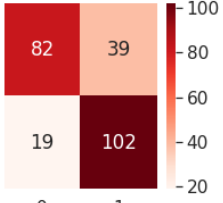
AB {'fit_time': 0.0071, 'score_time': 0.0064, 'test_balanced_accuracy':
0.7439, 'test_precision_weighted': 0.751, 'test_recall_weighted': 0.744,
'test_f1_weighted': 0.7421, 'test_roc_auc_ovo_weighted': 0.7439}

	precision	recall	f1-score	support		
0	0.75	0.73	0.74	121	0	88
1	0.74	0.76	0.75	121	1	33
accuracy			0.74	242		
macro avg	0.74	0.74	0.74	242		
weighted avg	0.74	0.74	0.74	242		



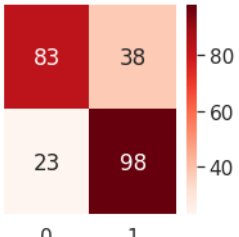
```
GB {'fit_time': 0.899, 'score_time': 0.0069, 'test_balanced_accuracy':
0.7606, 'test_precision_weighted': 0.7795, 'test_recall_weighted': 0.7608,
'test_f1_weighted': 0.7556, 'test_roc_auc_ovo_weighted': 0.8212}
```

	precision	recall	f1-score	support	
0	0.81	0.68	0.74	121	0
1	0.72	0.84	0.78	121	1
accuracy			0.76	242	
macro avg	0.77	0.76	0.76	242	
weighted avg	0.77	0.76	0.76	242	



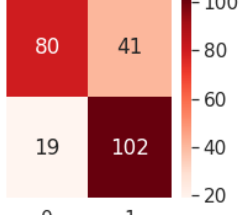
```
xgb {'fit_time': 0.178, 'score_time': 0.0068, 'test_balanced_accuracy':
0.7484, 'test_precision_weighted': 0.7705, 'test_recall_weighted': 0.7485,
'test_f1_weighted': 0.7417, 'test_roc_auc_ovo_weighted': 0.8397}
```

	precision	recall	f1-score	support	
0	0.78	0.69	0.73	121	0
1	0.72	0.81	0.76	121	1
accuracy			0.75	242	
macro avg	0.75	0.75	0.75	242	
weighted avg	0.75	0.75	0.75	242	



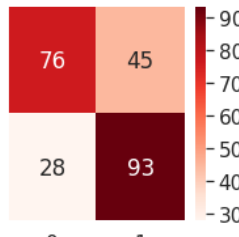
```
Bag_xgb {'fit_time': 0.8923, 'score_time': 0.0155,
'test_balanced_accuracy': 0.7526, 'test_precision_weighted': 0.769,
'test_recall_weighted': 0.7528, 'test_f1_weighted': 0.7474,
'test_roc_auc_ovo_weighted': 0.7967}
```

	precision	recall	f1-score	support	
0	0.81	0.66	0.73	121	0
1	0.71	0.84	0.77	121	1
accuracy			0.75	242	
macro avg	0.76	0.75	0.75	242	
weighted avg	0.76	0.75	0.75	242	



```
knn {'fit_time': 0.0007, 'score_time': 0.008, 'test_balanced_accuracy':
0.699, 'test_precision_weighted': 0.7153, 'test_recall_weighted': 0.6995,
'test_f1_weighted': 0.6886, 'test_roc_auc_ovo_weighted': 0.7764}
```

	precision	recall	f1-score	support	
0	0.73	0.63	0.68	121	0
1	0.67	0.77	0.72	121	1
accuracy			0.70	242	
macro avg	0.70	0.70	0.70	242	
weighted avg	0.70	0.70	0.70	242	



```

NB {'fit_time': 0.0015, 'score_time': 0.0077, 'test_balanced_accuracy':
0.7449, 'test_precision_weighted': 0.7549, 'test_recall_weighted': 0.7442,
'test_f1_weighted': 0.741, 'test_roc_auc_ovo_weighted': 0.8272}
precision    recall  f1-score   support

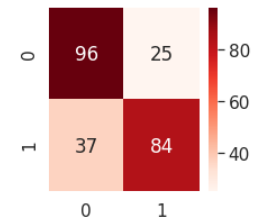
```

```

      0      0.72      0.79      0.76      121
      1      0.77      0.69      0.73      121

 accuracy      0.74      242
 macro avg      0.75      0.74      0.74      242
 weighted avg      0.75      0.74      0.74      242

```



```

svm {'fit_time': 0.0175, 'score_time': 0.0074, 'test_balanced_accuracy':
0.7654, 'test_precision_weighted': 0.7975, 'test_recall_weighted': 0.7657,
'test_f1_weighted': 0.7553, 'test_roc_auc_ovo_weighted': 0.8324}
precision    recall  f1-score   support

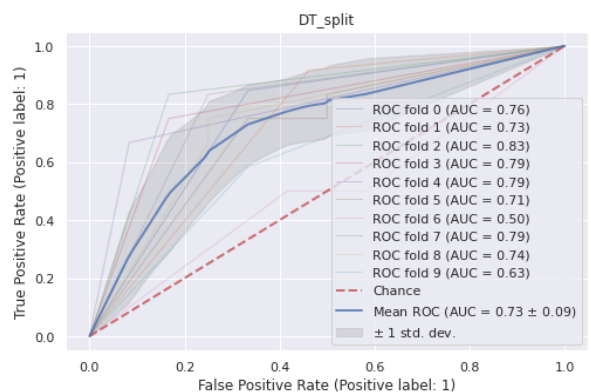
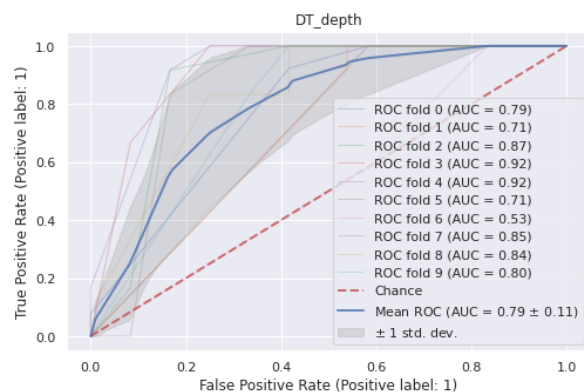
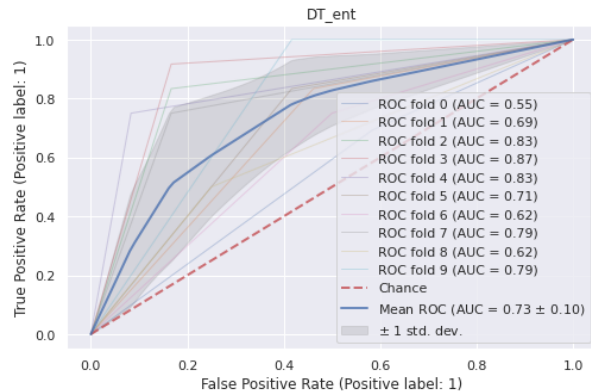
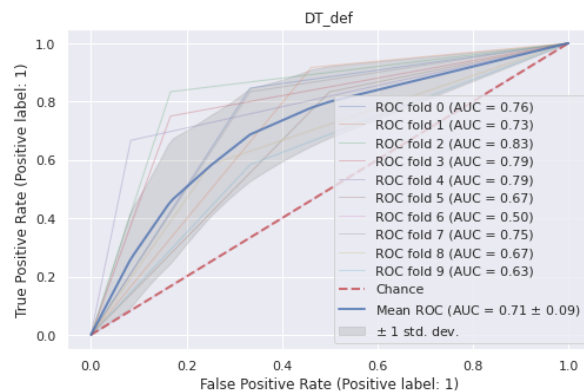
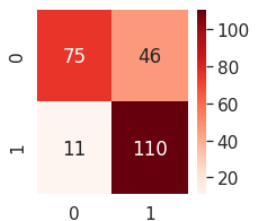
```

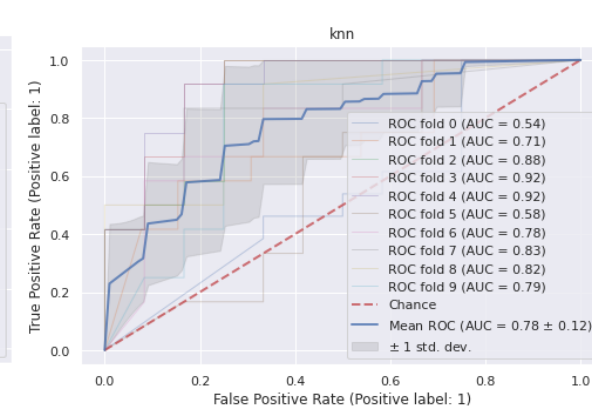
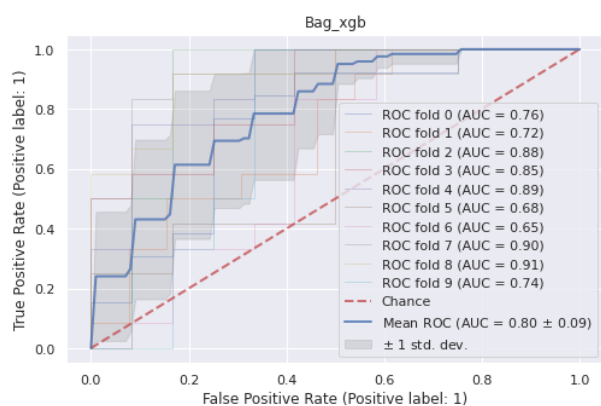
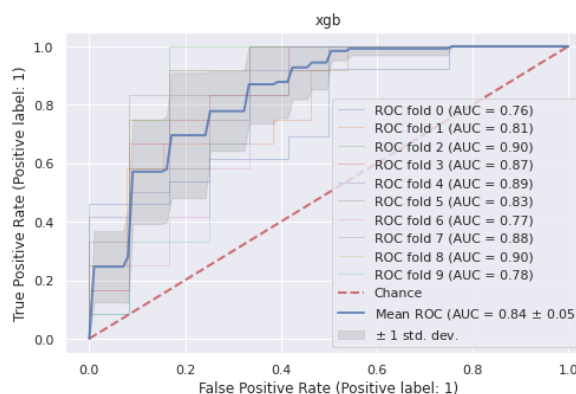
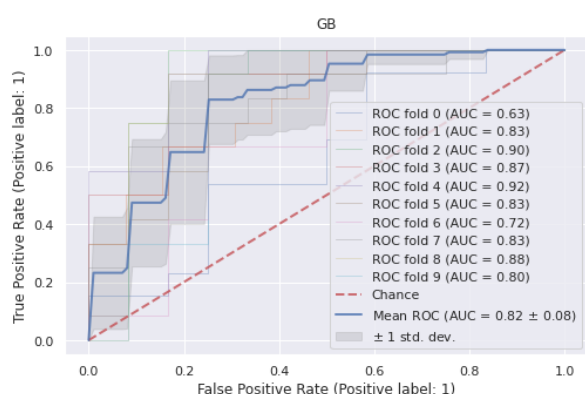
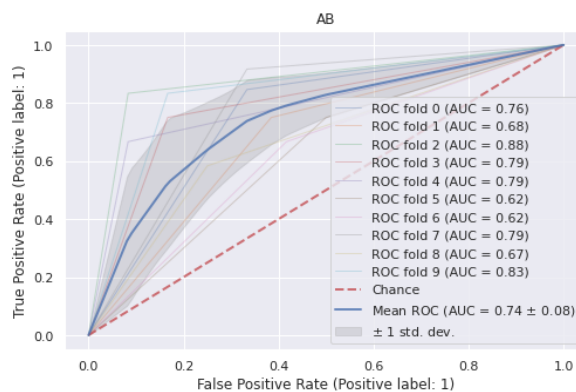
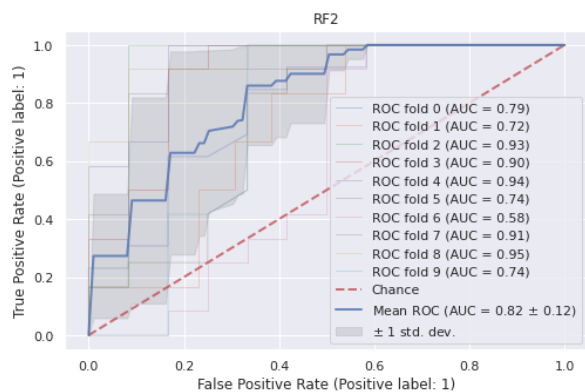
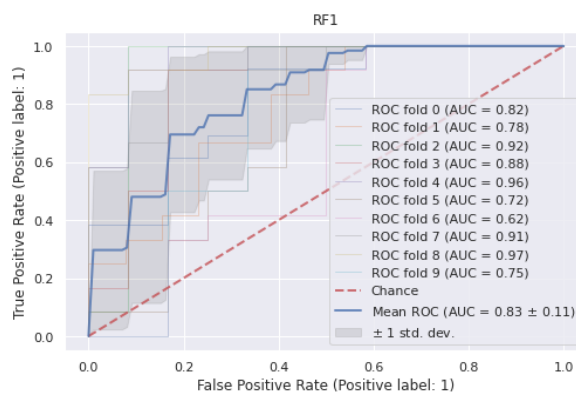
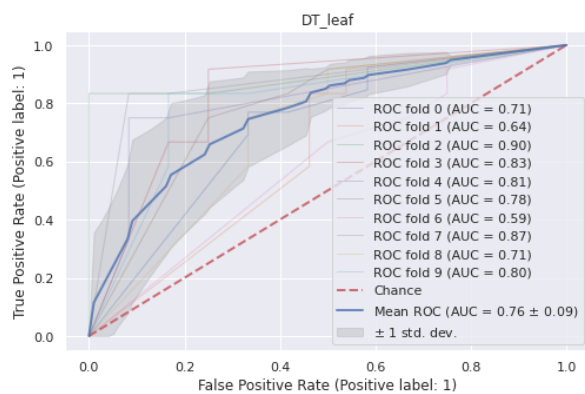
```

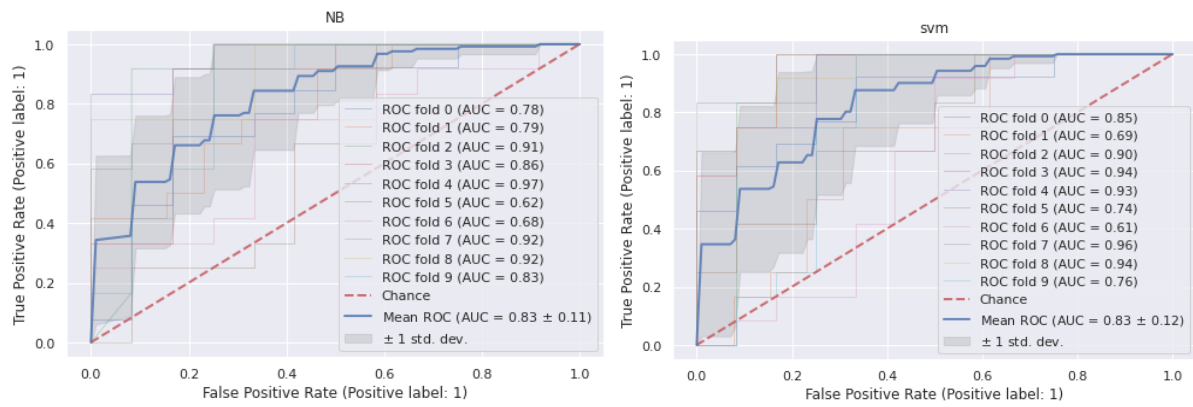
      0      0.87      0.62      0.72      121
      1      0.71      0.91      0.79      121

 accuracy      0.76      242
 macro avg      0.79      0.76      0.76      242
 weighted avg      0.79      0.76      0.76      242

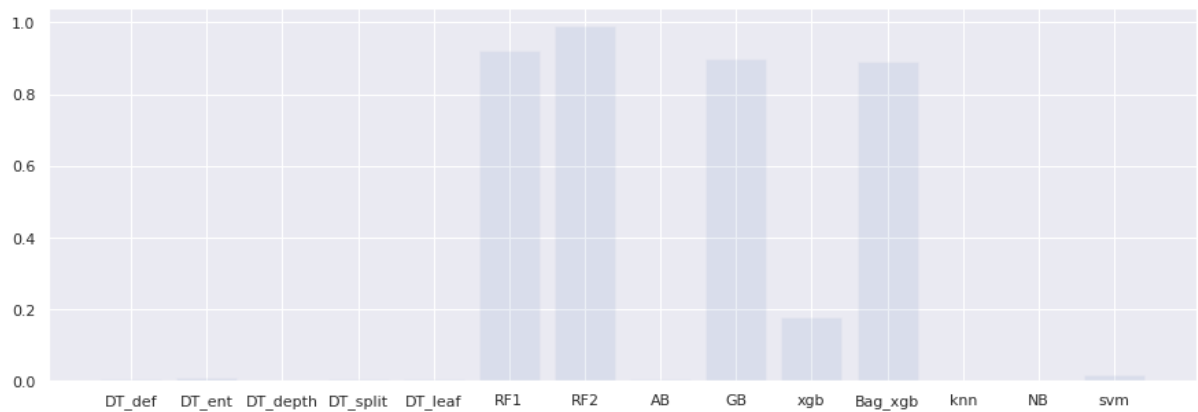
```



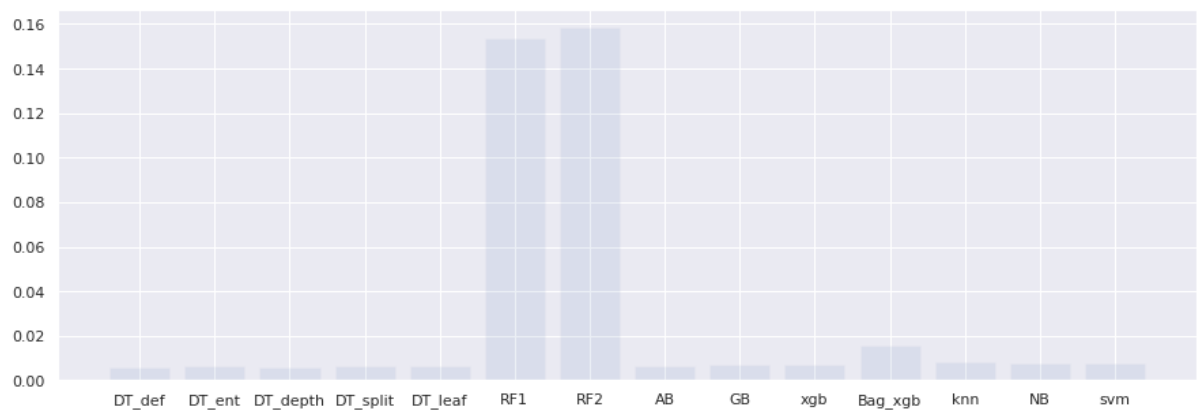




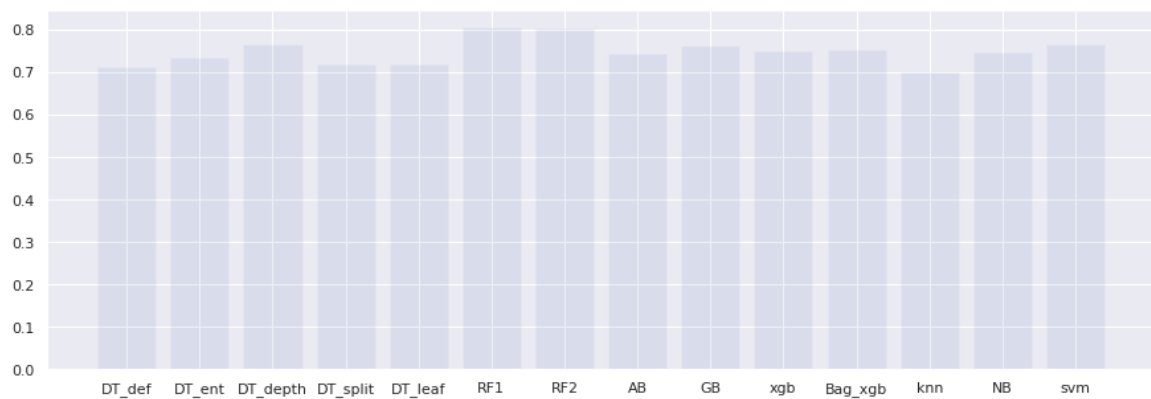
Comparison of fit_time feature of classification algorithms



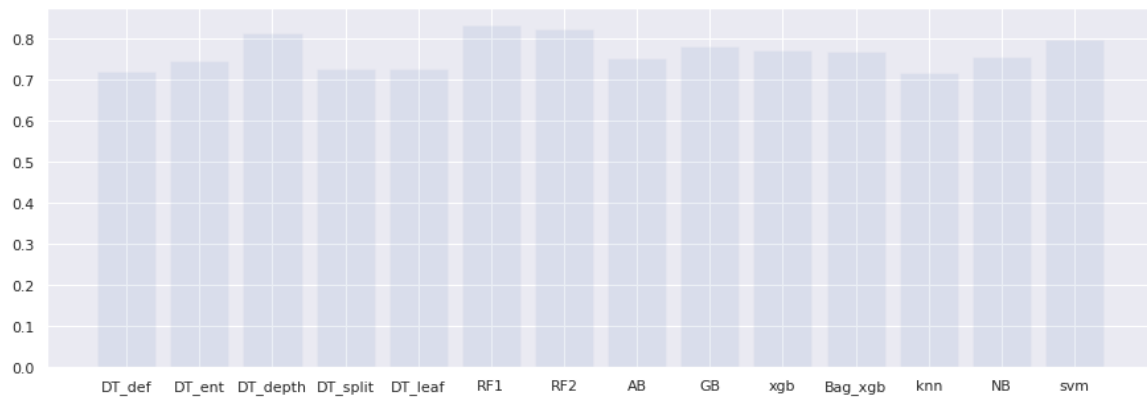
Comparison of score_time feature of classification algorithms



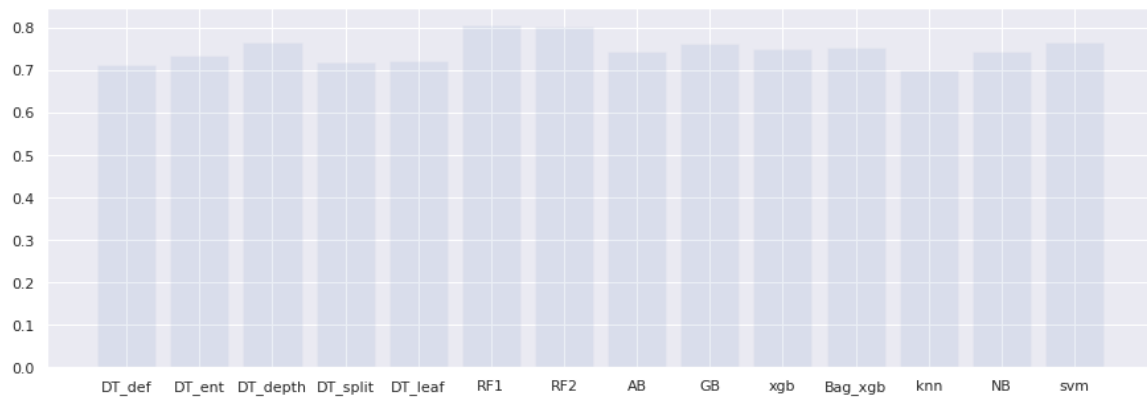
Comparison of test_balanced_accuracy feature of classification algorithms



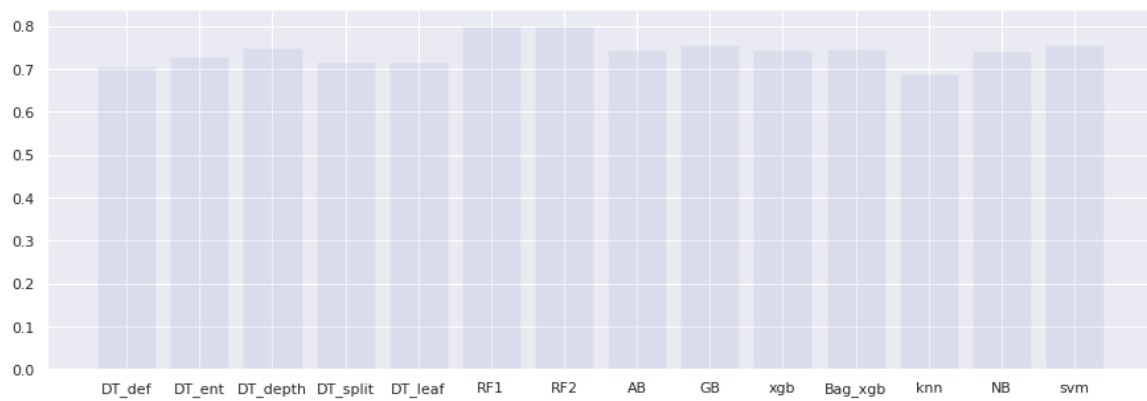
Comparison of test_precision_weighted feature of classification algorithms



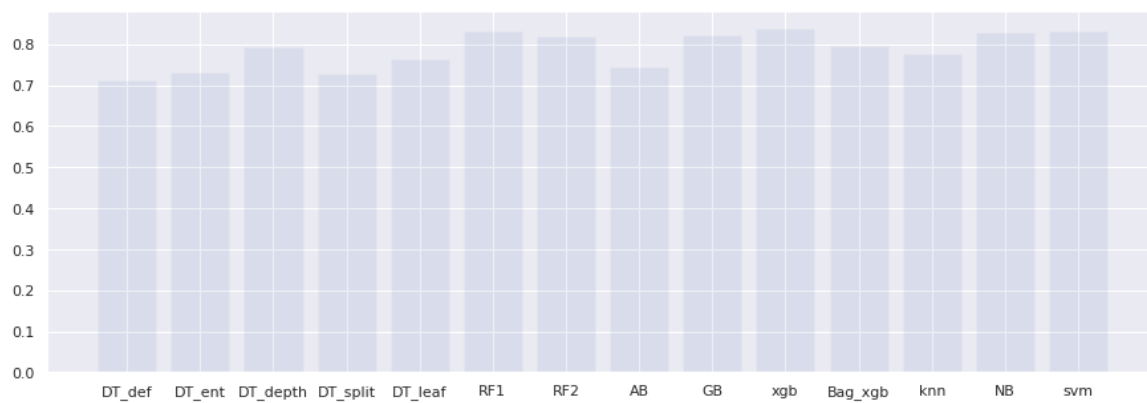
Comparison of test_recall_weighted feature of classification algorithms



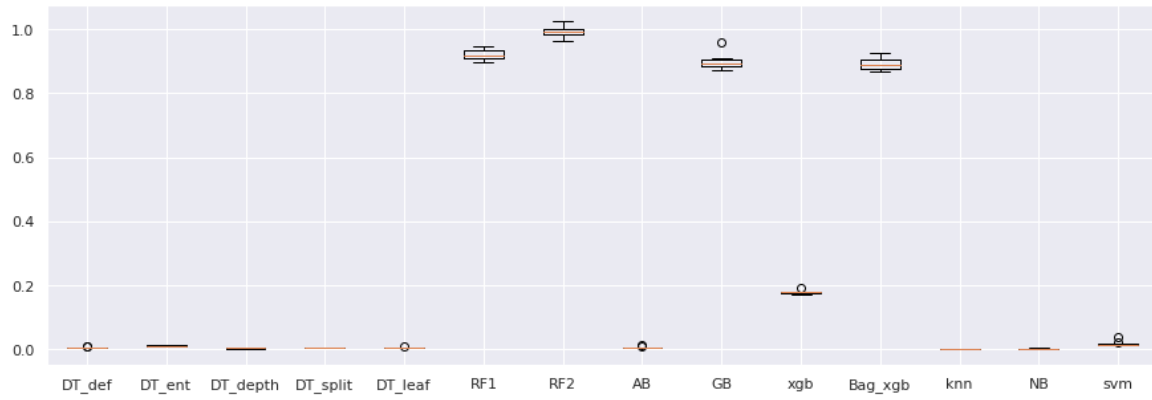
Comparison of test_f1_weighted feature of classification algorithms



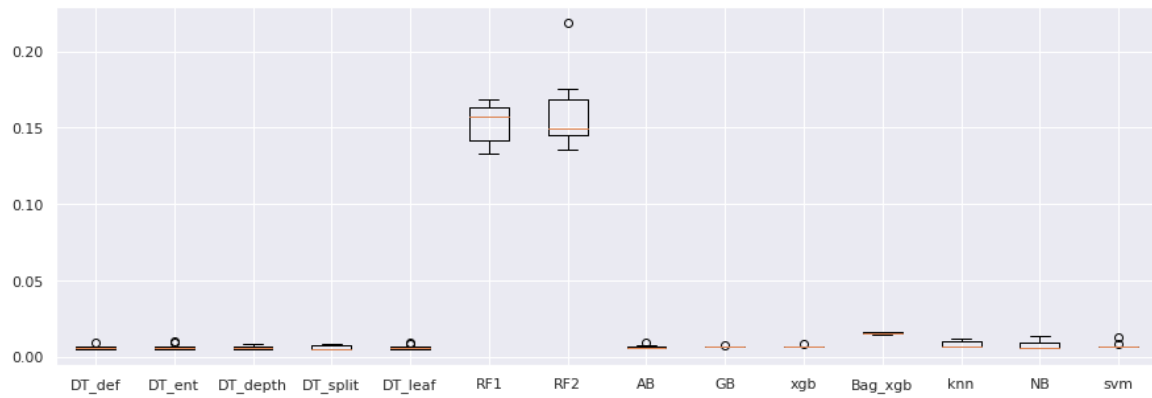
Comparison of test_roc_auc_ovo_weighted feature of classification algorithms



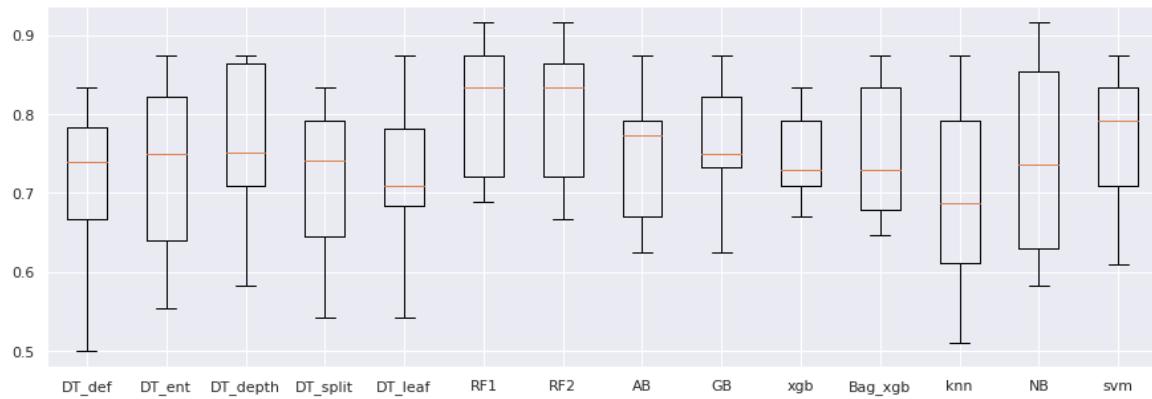
Comparison of score fit_time of classification algorithms



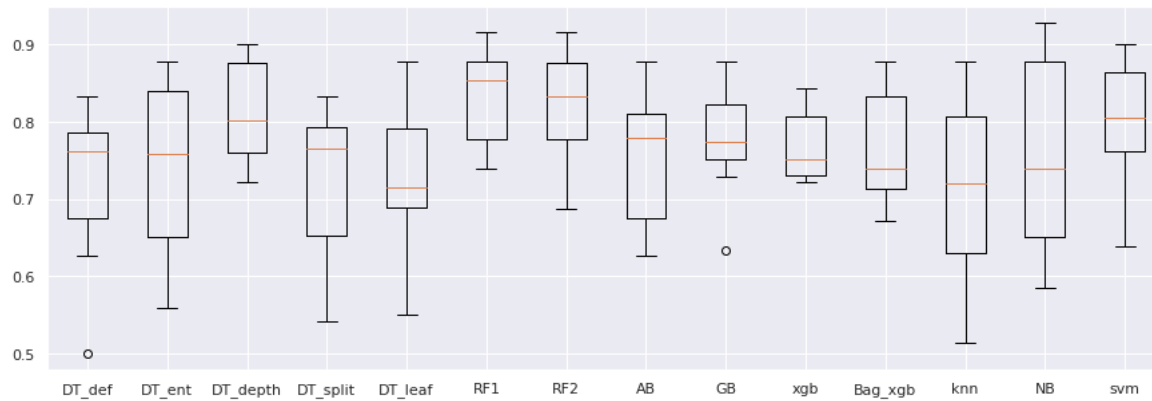
Comparison of score score_time of classification algorithms

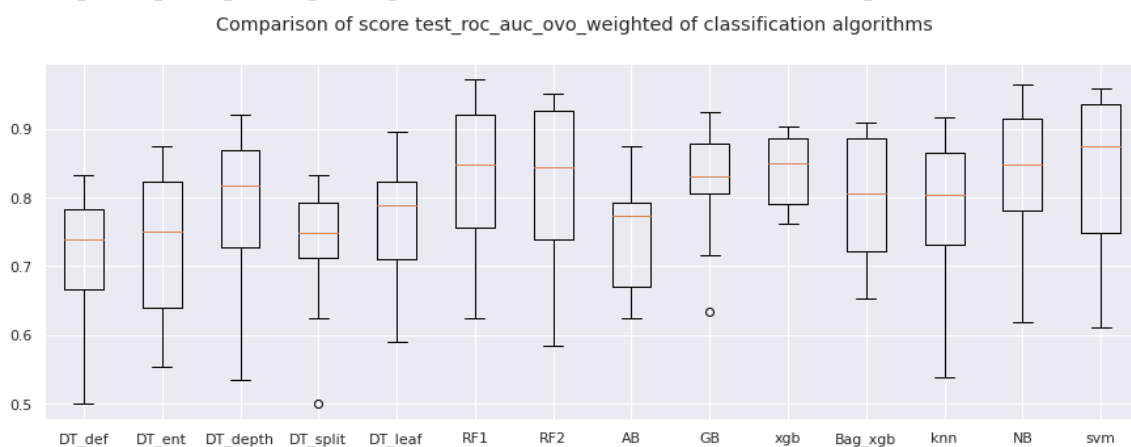
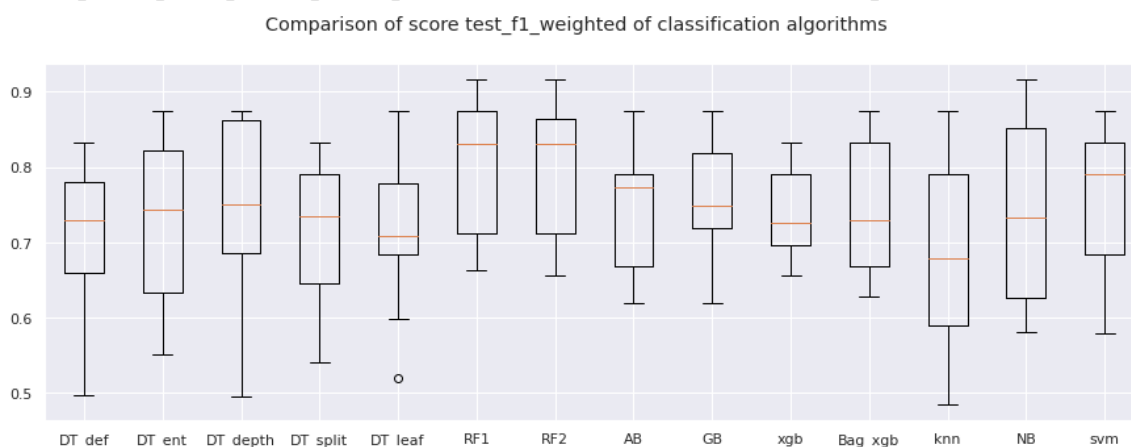
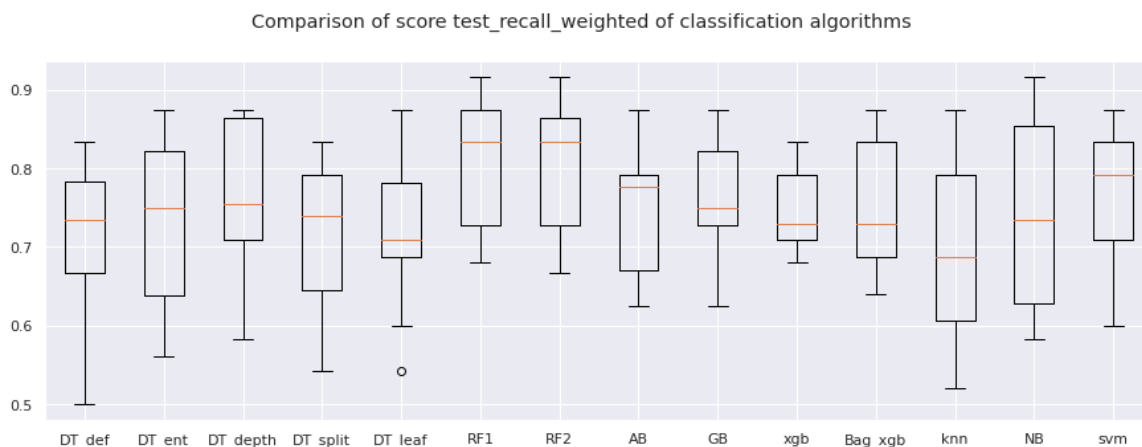


Comparison of score test_balanced_accuracy of classification algorithms

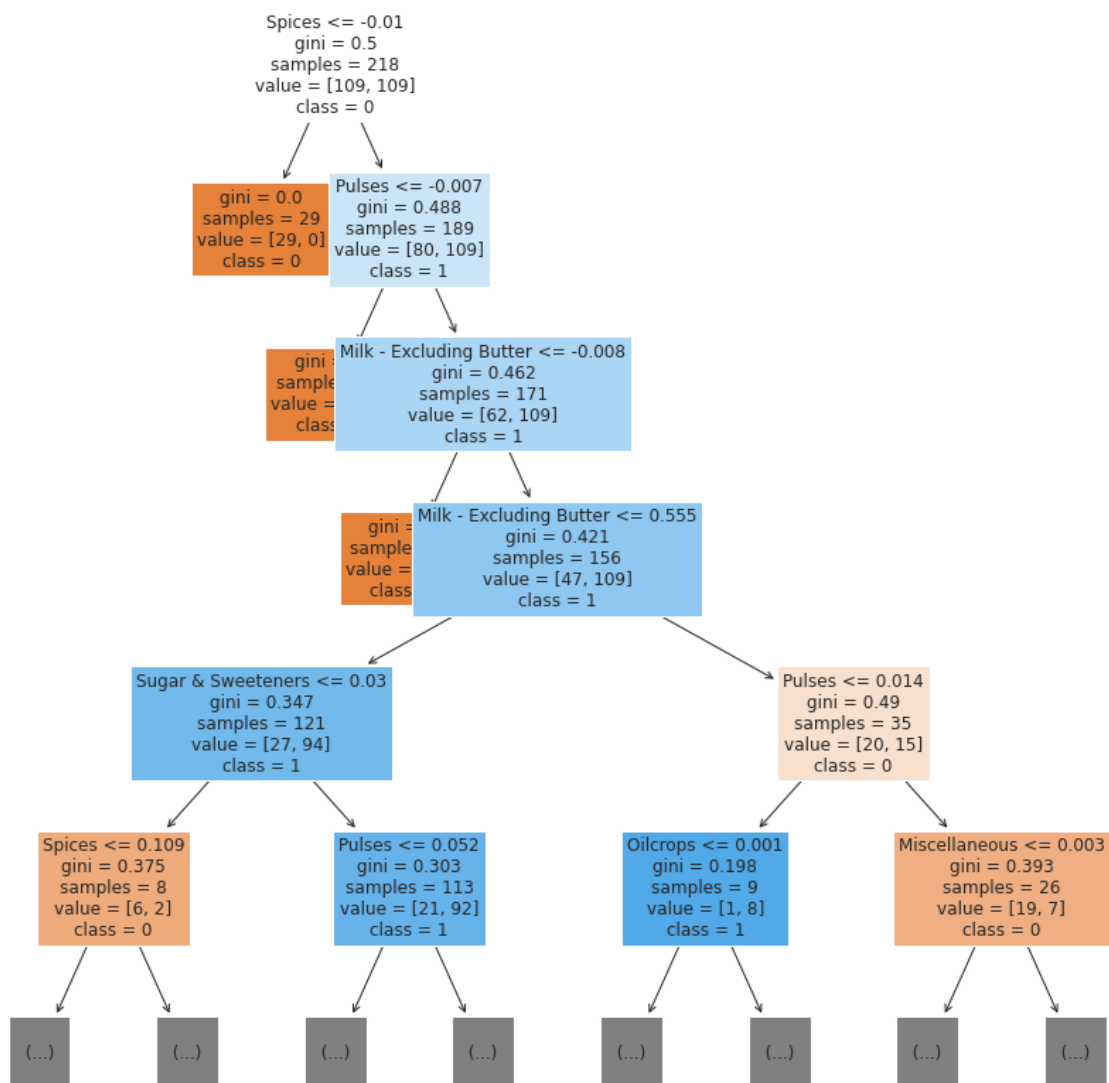


Comparison of score test_precision_weighted of classification algorithms





Z tohto reportu je ľahko vidieť, že najlepšie výsledky sa podarilo dosiahnuť pomocou metódy RandomForest RF1 (RF1 sa pri hľadaní najlepšieho rozdelenia pozerá na fixne daný počet atribútov, RF2 to má nastavené na auto), kde precision bol na úrovni 82%, avšak má najdlhší čas tréovania. Druhé najlepšie výsledky sa podarilo dosiahnuť pomocou DT_depth (rozhodovací strom s obmedzenou hĺbkou) a to precision na úrovni 80% a tretí najlepší výsledok sa podarilo získať pomocou SVM a to precision na úrovni 79%. Najhorší precision dosiahol KNN a to na úrovni 70%.



Výrez z rozhodovacieho stromu naznačuje, že syntetické generovanie dát pomocou MDO (dôjde k umelému dogenerovaniu záporných čísel) a strom vyzerá trochu degenerovane. Za významné atribúty môžeme považovať mliečne výrobky, cukor, strukoviny. Keďže sa viac osvedčil oversamplingový algoritmus ADASYN, tak v ďalších prípadoch budem používať už iba ten.

Target atribút: **3 skupiny (0 pre [0, 0.7), 1 pre [0.75, 0.85), 2 pre [0.85, 1])**

Pomer tried pred oversamplingom: **31:40:93**

Oversampling algoritmus: **ADASYN**

Algoritmy: DecisionTreeClassifier, RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier, XGBClassifier, BaggingClassifier(XGBClassifier), KNeighborsClassifier, GaussianNB, SVC

DT_def {'fit_time': 0.0059, 'score_time': 0.0116, 'test_balanced_accuracy': 0.6407, 'test_precision_weighted': 0.6413, 'test_recall_weighted': 0.6414, 'test_f1_weighted': 0.6289, 'test_roc_auc_ovo_weighted': 0.7308}

	precision	recall	f1-score	support				
0	0.70	0.79	0.74	91	0	72	9	10
1	0.62	0.66	0.64	92	1	10	61	21
2	0.59	0.47	0.52	93	2	21	28	44
accuracy			0.64	276				
macro avg	0.64	0.64	0.64	276				
weighted avg	0.64	0.64	0.64	276				

DT_ent {'fit_time': 0.0112, 'score_time': 0.0103, 'test_balanced_accuracy': 0.603, 'test_precision_weighted': 0.5903, 'test_recall_weighted': 0.6017, 'test_f1_weighted': 0.5734, 'test_roc_auc_ovo_weighted': 0.702}

	precision	recall	f1-score	support				
0	0.67	0.82	0.74	91	0	75	8	8
1	0.57	0.67	0.62	92	1	11	62	19
2	0.52	0.31	0.39	93	2	26	38	29
accuracy			0.60	276				
macro avg	0.59	0.60	0.58	276				
weighted avg	0.59	0.60	0.58	276				

DT_depth {'fit_time': 0.0037, 'score_time': 0.009, 'test_balanced_accuracy': 0.5537, 'test_precision_weighted': 0.5827, 'test_recall_weighted': 0.5548, 'test_f1_weighted': 0.5369, 'test_roc_auc_ovo_weighted': 0.7028}

	precision	recall	f1-score	support				
0	0.58	0.62	0.60	91	0	56	17	18
1	0.57	0.68	0.62	92	1	12	63	17
2	0.49	0.37	0.42	93	2	28	31	34
accuracy			0.55	276				
macro avg	0.55	0.56	0.55	276				
weighted avg	0.55	0.55	0.55	276				

DT_split {'fit_time': 0.0048, 'score_time': 0.0101, 'test_balanced_accuracy': 0.6381, 'test_precision_weighted': 0.6396, 'test_recall_weighted': 0.6377, 'test_f1_weighted': 0.6272, 'test_roc_auc_ovo_weighted': 0.7343}

	precision	recall	f1-score	support				
0	0.69	0.79	0.74	91	0	72	11	8
1	0.61	0.64	0.62	92	1	11	59	22
2	0.60	0.48	0.54	93	2	21	27	45
accuracy			0.64	276				
macro avg	0.63	0.64	0.63	276				
weighted avg	0.63	0.64	0.63	276				

DT_leaf {'fit_time': 0.0045, 'score_time': 0.0095,
'test_balanced_accuracy': 0.5904, 'test_precision_weighted': 0.5949,
'test_recall_weighted': 0.5898, 'test_f1_weighted': 0.5742,
'test_roc_auc_ovo_weighted': 0.7447}

	precision	recall	f1-score	support				
0	0.65	0.74	0.69	91	0	67	10	14
1	0.57	0.65	0.61	92	1	14	60	18
2	0.53	0.39	0.45	93	2	22	35	36
accuracy			0.59	276				
macro avg	0.58	0.59	0.58	276				
weighted avg	0.58	0.59	0.58	276				

RF1 {'fit_time': 0.9641, 'score_time': 0.1547, 'test_balanced_accuracy':
0.6926, 'test_precision_weighted': 0.7153, 'test_recall_weighted': 0.6926,
'test_f1_weighted': 0.6811, 'test_roc_auc_ovo_weighted': 0.8759}

	precision	recall	f1-score	support				
0	0.73	0.85	0.79	91	0	77	10	4
1	0.63	0.75	0.69	92	1	10	69	13
2	0.73	0.48	0.58	93	2	18	30	45
accuracy			0.69	276				
macro avg	0.70	0.69	0.68	276				
weighted avg	0.70	0.69	0.68	276				

RF2 {'fit_time': 1.0042, 'score_time': 0.1604, 'test_balanced_accuracy':
0.773, 'test_precision_weighted': 0.7838, 'test_recall_weighted': 0.7718,
'test_f1_weighted': 0.7631, 'test_roc_auc_ovo_weighted': 0.925}

	precision	recall	f1-score	support				
0	0.80	0.90	0.85	91	0	82	4	5
1	0.74	0.85	0.79	92	1	5	78	9
2	0.79	0.57	0.66	93	2	16	24	53
accuracy			0.77	276				
macro avg	0.77	0.77	0.77	276				
weighted avg	0.77	0.77	0.76	276				

AB {'fit_time': 0.0058, 'score_time': 0.0089, 'test_balanced_accuracy':
0.6304, 'test_precision_weighted': 0.6347, 'test_recall_weighted': 0.6306,
'test_f1_weighted': 0.6206, 'test_roc_auc_ovo_weighted': 0.7228}

	precision	recall	f1-score	support				
0	0.73	0.76	0.74	91	0	69	13	9
1	0.58	0.67	0.62	92	1	8	62	22
2	0.58	0.46	0.51	93	2	18	32	43
accuracy			0.63	276				
macro avg	0.63	0.63	0.63	276				
weighted avg	0.63	0.63	0.63	276				


```
GB {'fit_time': 3.3086, 'score_time': 0.012, 'test_balanced_accuracy':
0.7763, 'test_precision_weighted': 0.786, 'test_recall_weighted': 0.7757,
'test_f1_weighted': 0.7707, 'test_roc_auc_ovo_weighted': 0.9166}
```

	precision	recall	f1-score	support				
0	0.79	0.90	0.84	91	0	82	4	5
1	0.74	0.80	0.77	92	1	9	74	9
2	0.81	0.62	0.70	93	2	13	22	58
accuracy			0.78	276				
macro avg	0.78	0.78	0.77	276				
weighted avg	0.78	0.78	0.77	276				

```
xgb {'fit_time': 0.5649, 'score_time': 0.0129, 'test_balanced_accuracy':
0.7307, 'test_precision_weighted': 0.7323, 'test_recall_weighted': 0.7288,
'test_f1_weighted': 0.7178, 'test_roc_auc_ovo_weighted': 0.903}
```

	precision	recall	f1-score	support				
0	0.75	0.90	0.82	91	0	82	3	6
1	0.72	0.79	0.75	92	1	7	73	12
2	0.72	0.49	0.59	93	2	21	26	46
accuracy			0.73	276				
macro avg	0.73	0.73	0.72	276				
weighted avg	0.73	0.73	0.72	276				

```
Bag_xgb {'fit_time': 2.6954, 'score_time': 0.0419,
'test_balanced_accuracy': 0.6504, 'test_precision_weighted': 0.6593,
'test_recall_weighted': 0.6483, 'test_f1_weighted': 0.6276,
'test_roc_auc_ovo_weighted': 0.8459}
```

	precision	recall	f1-score	support				
0	0.68	0.90	0.77	91	0	82	6	3
1	0.62	0.63	0.62	92	1	15	58	19
2	0.64	0.42	0.51	93	2	24	30	39
accuracy			0.65	276				
macro avg	0.64	0.65	0.63	276				
weighted avg	0.64	0.65	0.63	276				

```
knn {'fit_time': 0.0007, 'score_time': 0.0102, 'test_balanced_accuracy':
0.7411, 'test_precision_weighted': 0.77, 'test_recall_weighted': 0.7392,
'test_f1_weighted': 0.6972, 'test_roc_auc_ovo_weighted': 0.861}
```

	precision	recall	f1-score	support				
0	0.70	0.99	0.82	91	0	90	1	0
1	0.73	0.91	0.81	92	1	6	84	2
2	0.94	0.32	0.48	93	2	33	30	30
accuracy			0.74	276				
macro avg	0.79	0.74	0.70	276				
weighted avg	0.79	0.74	0.70	276				

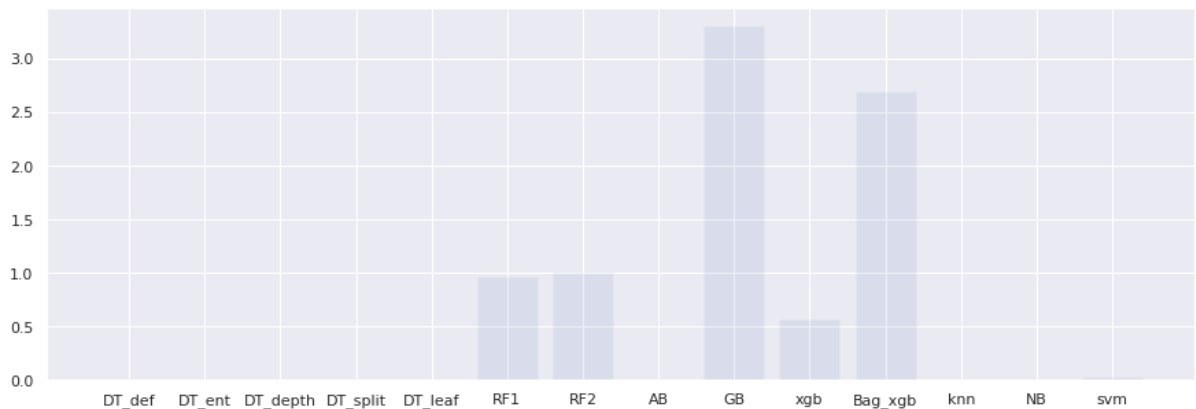
```
NB {'fit_time': 0.0014, 'score_time': 0.0109, 'test_balanced_accuracy':
0.4296, 'test_precision_weighted': 0.3865, 'test_recall_weighted': 0.427,
'test_f1_weighted': 0.3494, 'test_roc_auc_ovo_weighted': 0.6724}
```

	precision	recall	f1-score	support				
0	0.43	0.93	0.59	91	0	85	5	1
1	0.42	0.23	0.30	92	1	57	21	14
2	0.44	0.13	0.20	93	2	57	24	12
accuracy			0.43	276				
macro avg	0.43	0.43	0.36	276				
weighted avg	0.43	0.43	0.36	276				

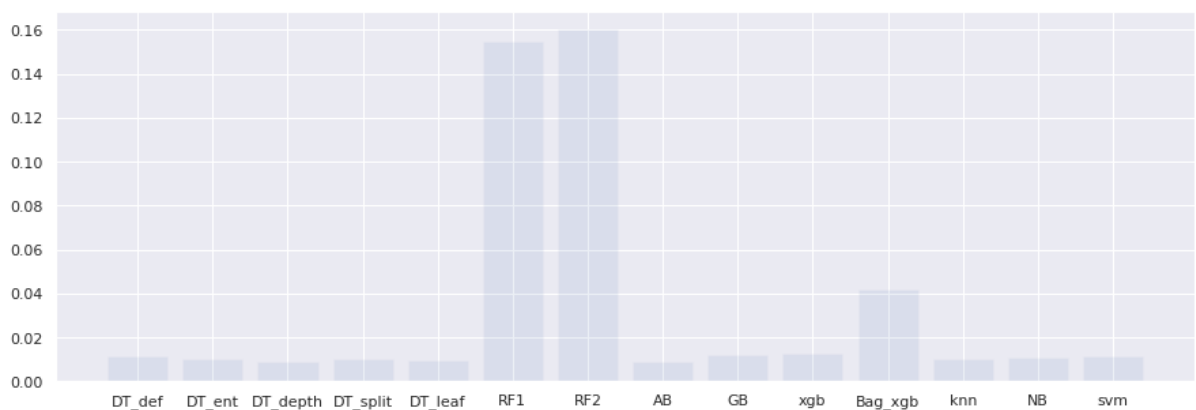
```
svm {'fit_time': 0.033, 'score_time': 0.0114, 'test_balanced_accuracy':
0.6037, 'test_precision_weighted': 0.6296, 'test_recall_weighted': 0.6015,
'test_f1_weighted': 0.5913, 'test_roc_auc_ovo_weighted': 0.7895}
```

	precision	recall	f1-score	support				
0	0.67	0.86	0.75	91	0	78	10	3
1	0.52	0.54	0.53	92	1	21	50	21
2	0.61	0.41	0.49	93	2	18	37	38
accuracy			0.60	276				
macro avg	0.60	0.60	0.59	276				
weighted avg	0.60	0.60	0.59	276				

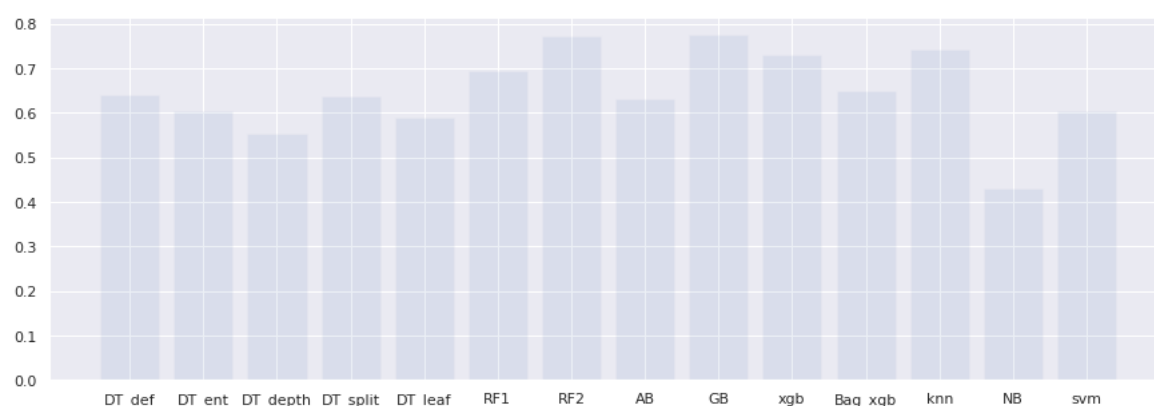
Comparison of fit_time feature of classification algorithms



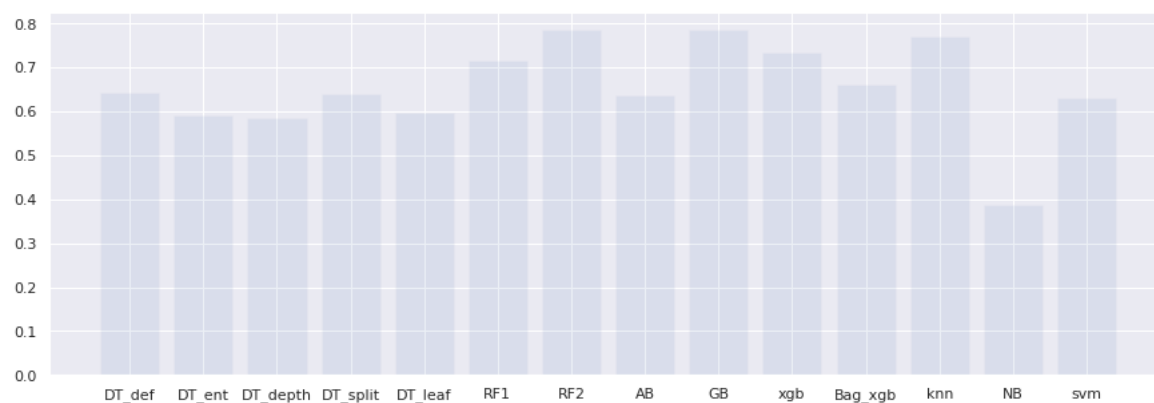
Comparison of score_time feature of classification algorithms



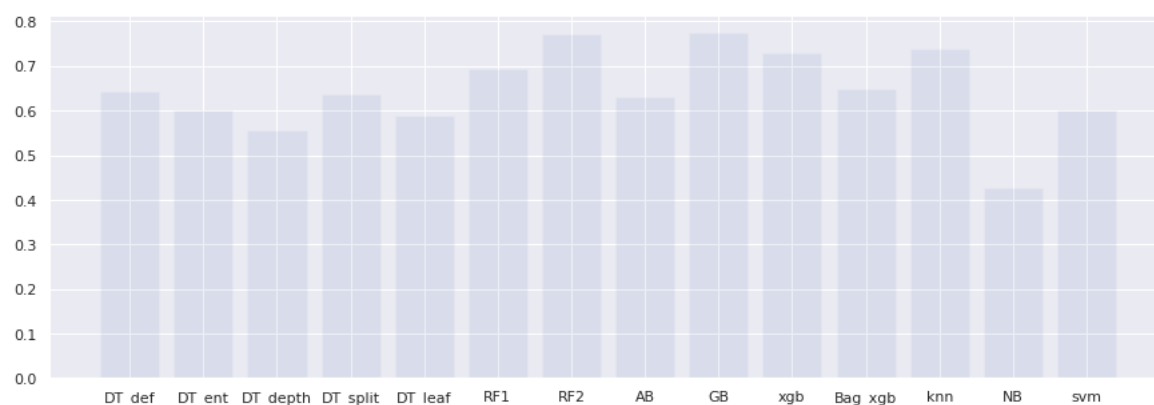
Comparison of test_balanced_accuracy feature of classification algorithms



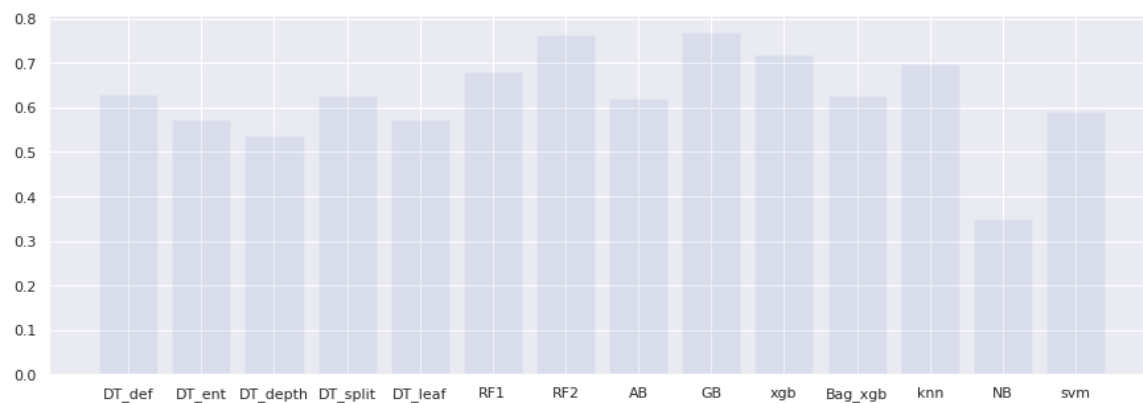
Comparison of test_precision_weighted feature of classification algorithms



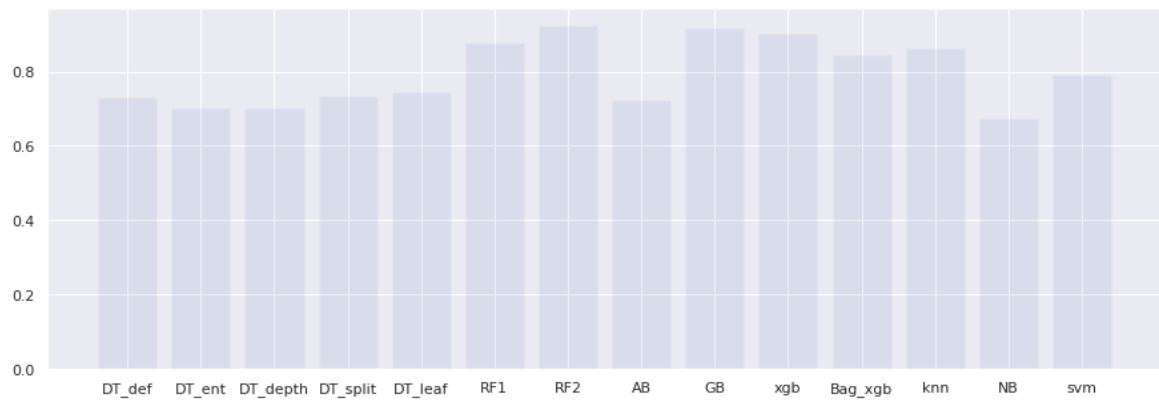
Comparison of test_recall_weighted feature of classification algorithms



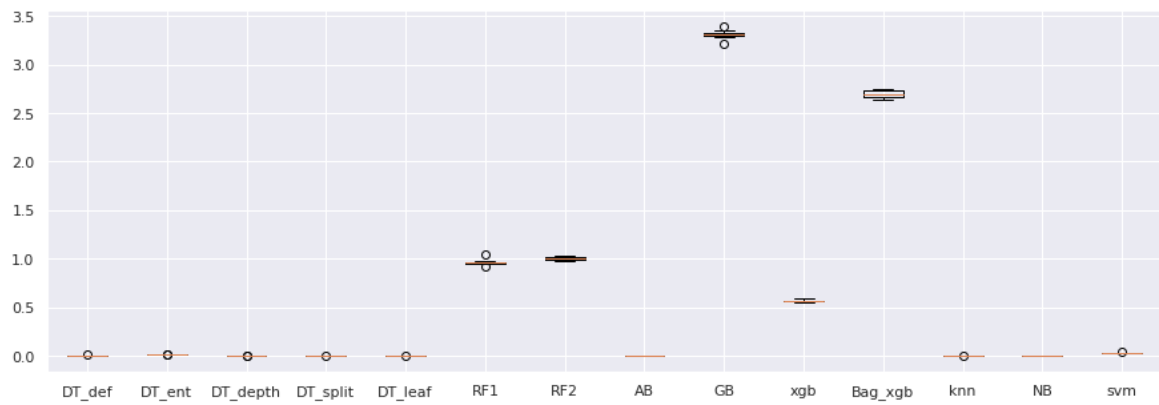
Comparison of test_f1_weighted feature of classification algorithms



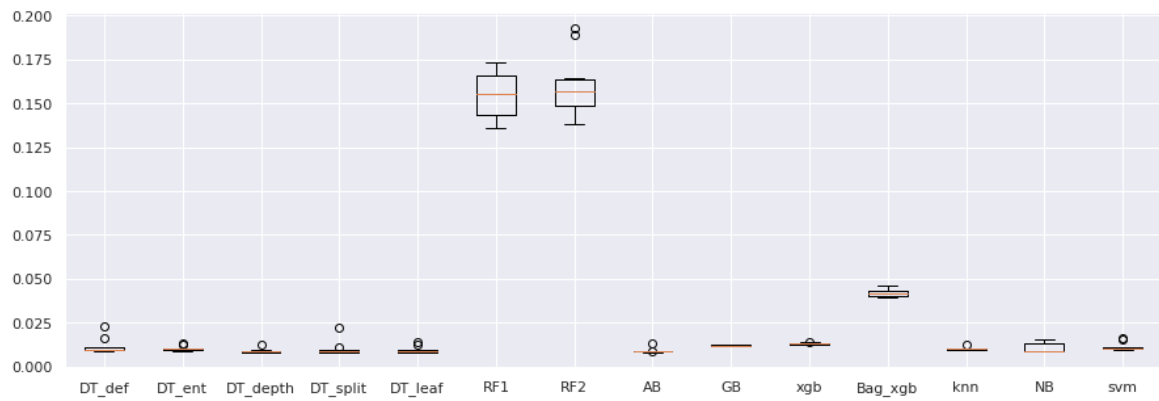
Comparison of test_roc_auc_ovo_weighted feature of classification algorithms



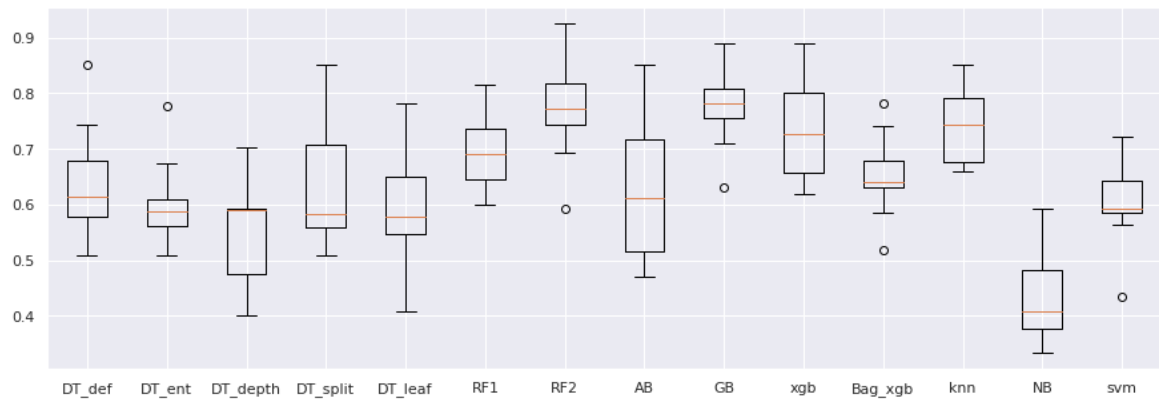
Comparison of score fit_time of classification algorithms



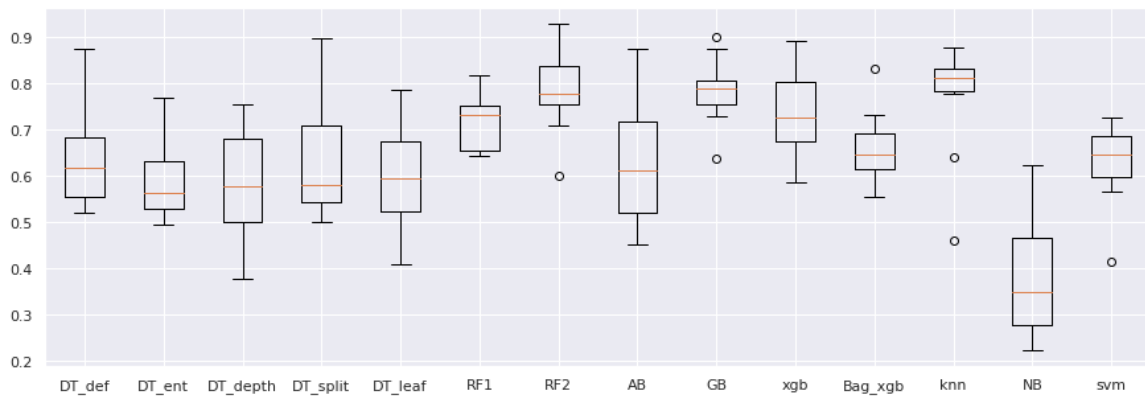
Comparison of score score_time of classification algorithms



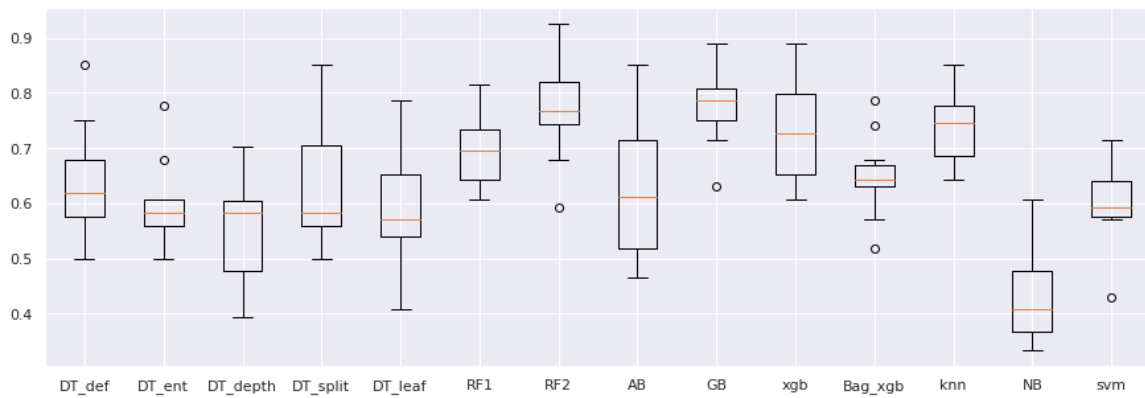
Comparison of score test_balanced_accuracy of classification algorithms



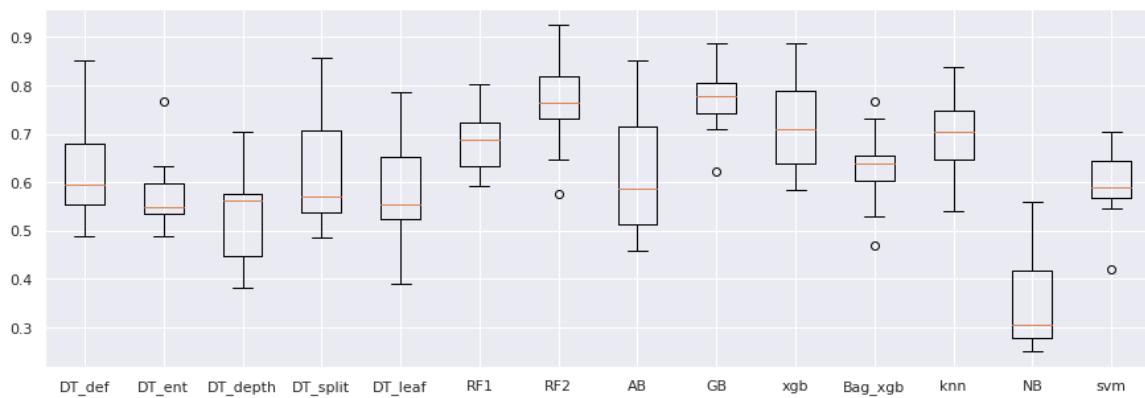
Comparison of score test_precision_weighted of classification algorithms



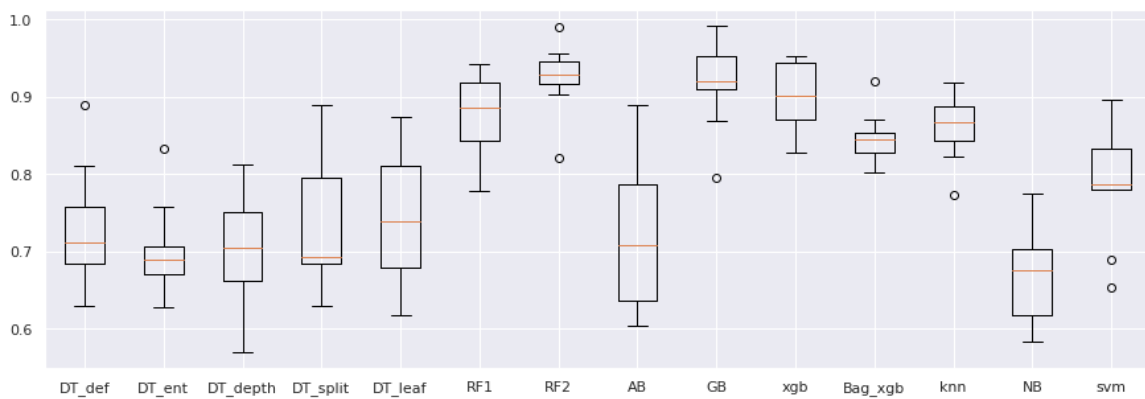
Comparison of score test_recall_weighted of classification algorithms

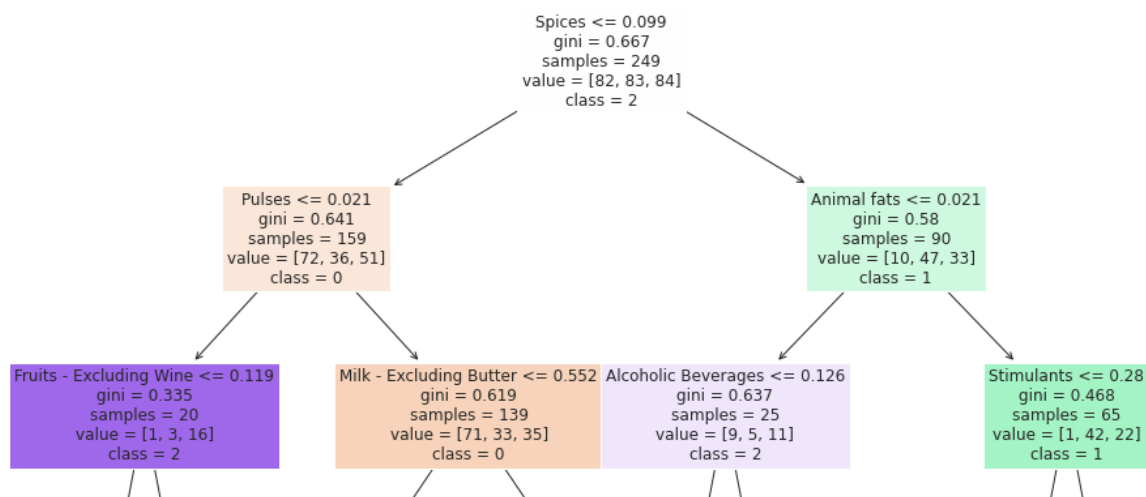


Comparison of score test_f1_weighted of classification algorithms



Comparison of score test_roc_auc_ovo_weighted of classification algorithms





V prípade keď cieľový atribút rozdelím do 3 skupín je ľahko vidieť, že najlepšie výsledky sa podarilo dosiahnuť pomocou metódy RandomForest RF2, kde precision bol na úrovni 78%, avšak má najdlhší čas trénovania. Rovnako dobrý precision dosiahol GradientBoosting GB. Druhé najlepšie výsledky sa podarilo dosiahnuť pomocou KNN a to precision na úrovni 77%. Najhorší precision dosiahol NaivnýBayes a to na úrovni 38%. Celkovo je vidieť väčšie rozdiely medzi jednotlivými algoritmami.

Z výřexu stromu je vidieť, že za významné atribúty môžeme považovať mliečne výrobky, strukoviny, ovocie, korenie.

Target atribút: **4 skupiny (0 pre [0, 0.7), 1 pre [0.7, 0.8), 2 pre [0.8, 0.9), 3 pre [0.9, 1])**

Pomer tried pred oversamplingom: **31:23:45:65**

Oversampling algoritmus: **ADASYN**

Algoritmy: DecisionTreeClassifier, RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier, XGBClassifier, BaggingClassifier(XGBClassifier), KNeighborsClassifier, GaussianNB, SVC

DT_def {'fit_time': 0.0046, 'score_time': 0.0151, 'test_balanced_accuracy': 0.4673, 'test_precision_weighted': 0.4663, 'test_recall_weighted': 0.4748, 'test_f1_weighted': 0.4557, 'test_roc_auc_ovo_weighted': 0.6468}

	precision	recall	f1-score	support					
0	0.54	0.58	0.56	64	0	37	7	9	11
1	0.57	0.60	0.59	68	1	10	41	7	10
2	0.39	0.37	0.38	52	2	7	8	19	18
3	0.35	0.32	0.34	65	3	14	16	14	21
						0	1	2	3
accuracy			0.47	249					
macro avg	0.46	0.47	0.46	249					
weighted avg	0.47	0.47	0.47	249					

DT_ent {'fit_time': 0.0104, 'score_time': 0.0146, 'test_balanced_accuracy': 0.4933, 'test_precision_weighted': 0.5012, 'test_recall_weighted': 0.5017, 'test_f1_weighted': 0.4934, 'test_roc_auc_ovo_weighted': 0.6639}

	precision	recall	f1-score	support					
0	0.58	0.59	0.59	64	0	38	6	8	12
1	0.65	0.66	0.66	68	1	6	45	8	9
2	0.36	0.33	0.34	52	2	5	8	17	22
3	0.37	0.38	0.38	65	3	16	10	14	25
						0	1	2	3
accuracy			0.50	249					
macro avg	0.49	0.49	0.49	249					
weighted avg	0.50	0.50	0.50	249					

DT_depth {'fit_time': 0.0041, 'score_time': 0.0182, 'test_balanced_accuracy': 0.4235, 'test_precision_weighted': 0.4592, 'test_recall_weighted': 0.4345, 'test_f1_weighted': 0.4147, 'test_roc_auc_ovo_weighted': 0.6503}

	precision	recall	f1-score	support					
0	0.51	0.56	0.53	64	0	36	10	2	16
1	0.48	0.56	0.52	68	1	12	38	7	11
2	0.38	0.25	0.30	52	2	7	15	13	17
3	0.32	0.32	0.32	65	3	16	16	12	21
						0	1	2	3
accuracy			0.43	249					
macro avg	0.42	0.42	0.42	249					
weighted avg	0.43	0.43	0.43	249					

DT_split {'fit_time': 0.005, 'score_time': 0.0136, 'test_balanced_accuracy': 0.4762, 'test_precision_weighted': 0.4653, 'test_recall_weighted': 0.4827, 'test_f1_weighted': 0.46, 'test_roc_auc_ovo_weighted': 0.6619}

	precision	recall	f1-score	support					
0	0.56	0.62	0.59	64	0	40	8	8	8
1	0.54	0.57	0.56	68	1	12	39	8	9
2	0.39	0.37	0.38	52	2	7	9	19	17
3	0.39	0.34	0.36	65	3	13	16	14	22
						0	1	2	3
accuracy			0.48	249					
macro avg	0.47	0.48	0.47	249					
weighted avg	0.47	0.48	0.48	249					

DT_leaf {'fit_time': 0.0039, 'score_time': 0.014, 'test_balanced_accuracy': 0.4444, 'test_precision_weighted': 0.4359, 'test_recall_weighted': 0.4503, 'test_f1_weighted': 0.4323, 'test_roc_auc_ovo_weighted': 0.6746}

	precision	recall	f1-score	support					
0	0.50	0.58	0.54	64	0	37	7	7	13
1	0.52	0.56	0.54	68	1	13	38	9	8
2	0.38	0.35	0.36	52	2	7	13	18	14
3	0.35	0.29	0.32	65	3	17	15	14	19
accuracy			0.45	249					
macro avg	0.44	0.44	0.44	249					
weighted avg	0.44	0.45	0.44	249					

RF1 {'fit_time': 0.9374, 'score_time': 0.1553, 'test_balanced_accuracy': 0.567, 'test_precision_weighted': 0.6308, 'test_recall_weighted': 0.5863, 'test_f1_weighted': 0.5625, 'test_roc_auc_ovo_weighted': 0.8284}

	precision	recall	f1-score	support					
0	0.61	0.73	0.67	64	0	47	7	1	9
1	0.62	0.82	0.71	68	1	7	56	1	4
2	0.61	0.27	0.37	52	2	11	10	14	17
3	0.49	0.45	0.47	65	3	12	17	7	29
accuracy			0.59	249					
macro avg	0.58	0.57	0.55	249					
weighted avg	0.58	0.59	0.57	249					

RF2 {'fit_time': 1.0142, 'score_time': 0.1603, 'test_balanced_accuracy': 0.6496, 'test_precision_weighted': 0.6748, 'test_recall_weighted': 0.6623, 'test_f1_weighted': 0.6451, 'test_roc_auc_ovo_weighted': 0.8822}

	precision	recall	f1-score	support					
0	0.68	0.83	0.75	64	0	53	2	2	7
1	0.77	0.90	0.83	68	1	4	61	1	2
2	0.57	0.44	0.50	52	2	8	6	23	15
3	0.54	0.43	0.48	65	3	13	10	14	28
accuracy			0.66	249					
macro avg	0.64	0.65	0.64	249					
weighted avg	0.65	0.66	0.65	249					

AB {'fit_time': 0.0056, 'score_time': 0.0119, 'test_balanced_accuracy': 0.526, 'test_precision_weighted': 0.5314, 'test_recall_weighted': 0.5312, 'test_f1_weighted': 0.513, 'test_roc_auc_ovo_weighted': 0.6845}

	precision	recall	f1-score	support					
0	0.57	0.67	0.62	64	0	43	6	6	9
1	0.60	0.62	0.61	68	1	11	42	4	11
2	0.53	0.46	0.49	52	2	4	8	24	16
3	0.39	0.35	0.37	65	3	17	14	11	23
accuracy			0.53	249					
macro avg	0.52	0.53	0.52	249					
weighted avg	0.52	0.53	0.53	249					


```
GB {'fit_time': 4.1039, 'score_time': 0.017, 'test_balanced_accuracy':
0.605, 'test_precision_weighted': 0.6177, 'test_recall_weighted': 0.618,
'test_f1_weighted': 0.6033, 'test_roc_auc_ovo_weighted': 0.844}
```

	precision	recall	f1-score	support						
0	0.73	0.72	0.72	64	0	46	3	3	12	50
1	0.71	0.85	0.77	68	1	2	58	4	4	40
2	0.52	0.44	0.48	52	2	6	6	23	17	30
3	0.45	0.42	0.43	65	3	9	15	14	27	20
accuracy			0.62	249	3					10
macro avg	0.60	0.61	0.60	249		0	1	2	3	
weighted avg	0.61	0.62	0.61	249						

```
xgb {'fit_time': 0.6327, 'score_time': 0.0175, 'test_balanced_accuracy':
0.6199, 'test_precision_weighted': 0.6293, 'test_recall_weighted': 0.6303,
'test_f1_weighted': 0.6099, 'test_roc_auc_ovo_weighted': 0.8494}
```

	precision	recall	f1-score	support						
0	0.68	0.75	0.71	64	0	48	8	4	4	50
1	0.70	0.82	0.76	68	1	3	56	4	5	40
2	0.52	0.46	0.49	52	2	7	7	24	14	30
3	0.56	0.45	0.50	65	3	13	9	14	29	20
accuracy			0.63	249	3					10
macro avg	0.61	0.62	0.61	249		0	1	2	3	
weighted avg	0.62	0.63	0.62	249						

```
Bag_xgb {'fit_time': 2.951, 'score_time': 0.051, 'test_balanced_accuracy':
0.5398, 'test_precision_weighted': 0.5238, 'test_recall_weighted': 0.5587,
'test_f1_weighted': 0.52, 'test_roc_auc_ovo_weighted': 0.7939}
```

	precision	recall	f1-score	support						
0	0.61	0.73	0.67	64	0	47	4	5	8	50
1	0.61	0.79	0.69	68	1	7	54	3	4	40
2	0.42	0.27	0.33	52	2	11	13	14	14	30
3	0.48	0.37	0.42	65	3	12	18	11	24	20
accuracy			0.56	249	3					10
macro avg	0.53	0.54	0.53	249		0	1	2	3	
weighted avg	0.54	0.56	0.54	249						

```
knn {'fit_time': 0.0006, 'score_time': 0.0134, 'test_balanced_accuracy':
0.5898, 'test_precision_weighted': 0.6075, 'test_recall_weighted': 0.6065,
'test_f1_weighted': 0.5544, 'test_roc_auc_ovo_weighted': 0.8074}
```

	precision	recall	f1-score	support						
0	0.57	0.94	0.71	64	0	60	2	0	2	60
1	0.70	0.90	0.79	68	1	5	61	2	0	50
2	0.53	0.33	0.40	52	2	17	8	17	10	40
3	0.52	0.20	0.29	65	3	23	16	13	13	30
accuracy			0.61	249	3					20
macro avg	0.58	0.59	0.55	249		0	1	2	3	10
weighted avg	0.59	0.61	0.56	249						0

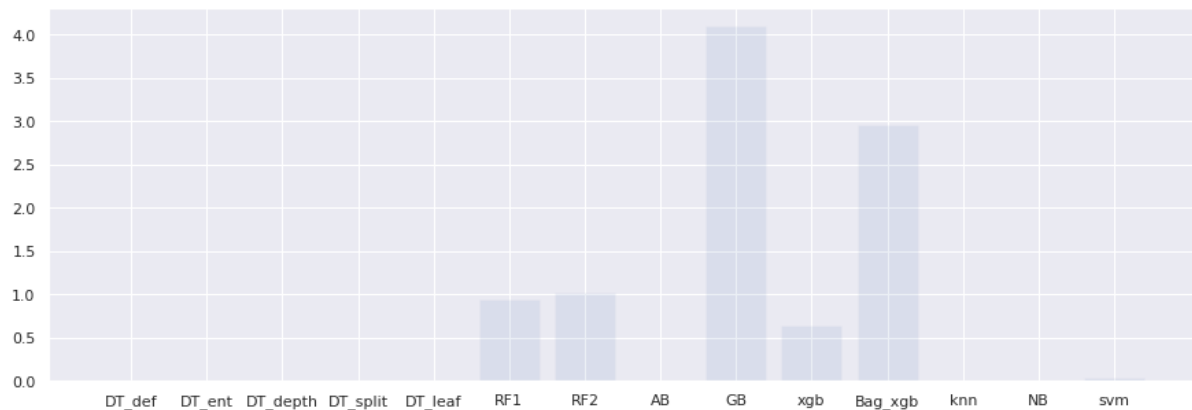
```
NB {'fit_time': 0.0016, 'score_time': 0.0152, 'test_balanced_accuracy':
0.3942, 'test_precision_weighted': 0.4426, 'test_recall_weighted': 0.4098,
'test_f1_weighted': 0.3508, 'test_roc_auc_ovo_weighted': 0.6272}
```

	precision	recall	f1-score	support					
0	0.36	0.94	0.52	64	0	60	2	0	2
1	0.57	0.35	0.44	68	1	34	24	3	7
2	0.30	0.06	0.10	52	2	38	6	3	5
3	0.52	0.23	0.32	65	3	36	10	4	15
accuracy			0.41	249					
macro avg	0.44	0.39	0.34	249					
weighted avg	0.45	0.41	0.36	249					

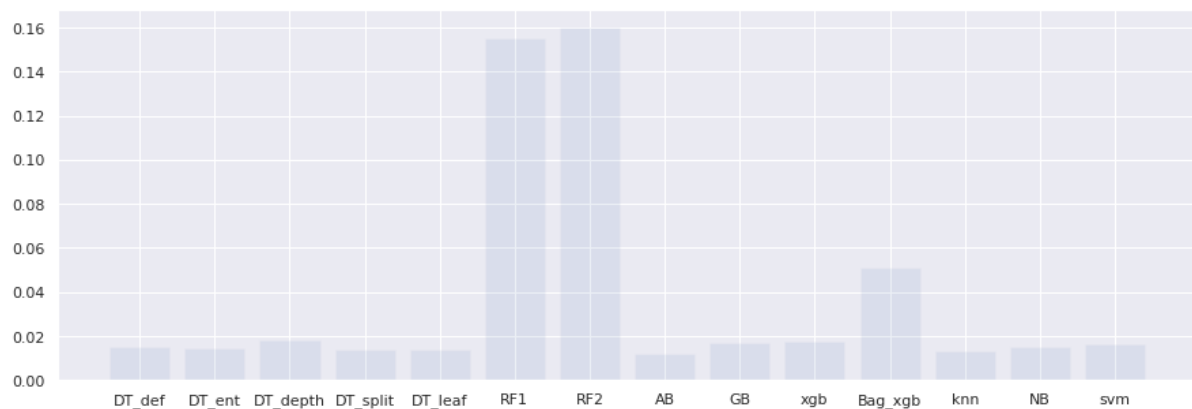
```
svm {'fit_time': 0.034, 'score_time': 0.0164, 'test_balanced_accuracy':
0.4624, 'test_precision_weighted': 0.4444, 'test_recall_weighted': 0.4782,
'test_f1_weighted': 0.4453, 'test_roc_auc_ovo_weighted': 0.7766}
```

	precision	recall	f1-score	support					
0	0.50	0.67	0.57	64	0	43	4	6	11
1	0.56	0.68	0.61	68	1	16	46	1	5
2	0.41	0.17	0.24	52	2	11	10	9	22
3	0.36	0.32	0.34	65	3	16	22	6	21
accuracy			0.48	249					
macro avg	0.46	0.46	0.44	249					
weighted avg	0.46	0.48	0.45	249					

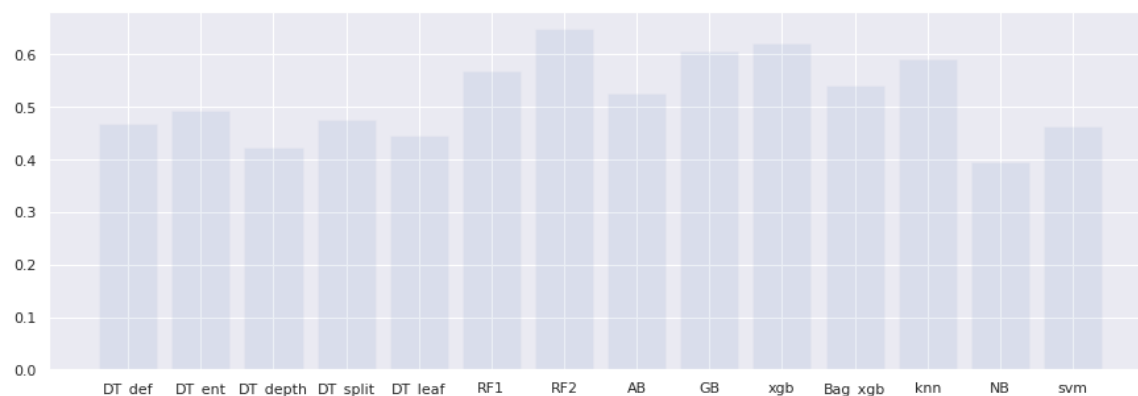
Comparison of fit_time feature of classification algorithms



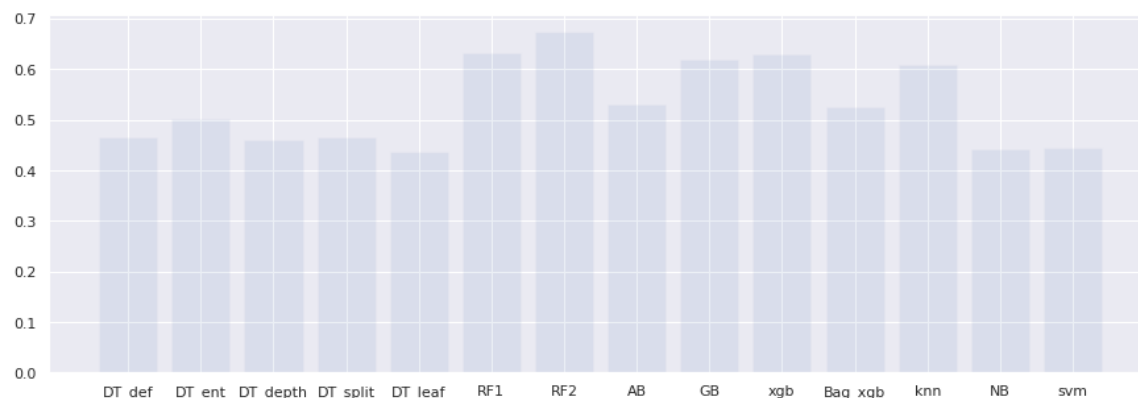
Comparison of score_time feature of classification algorithms



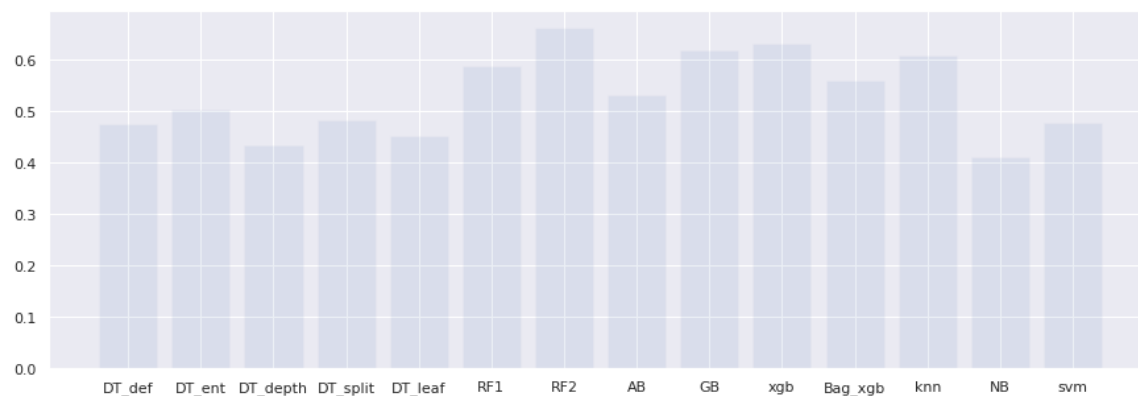
Comparison of test_balanced_accuracy feature of classification algorithms



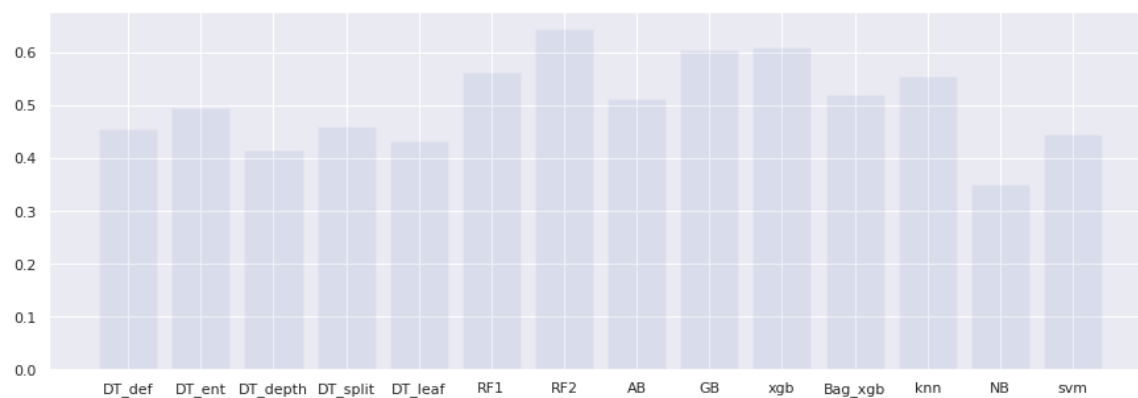
Comparison of test_precision_weighted feature of classification algorithms



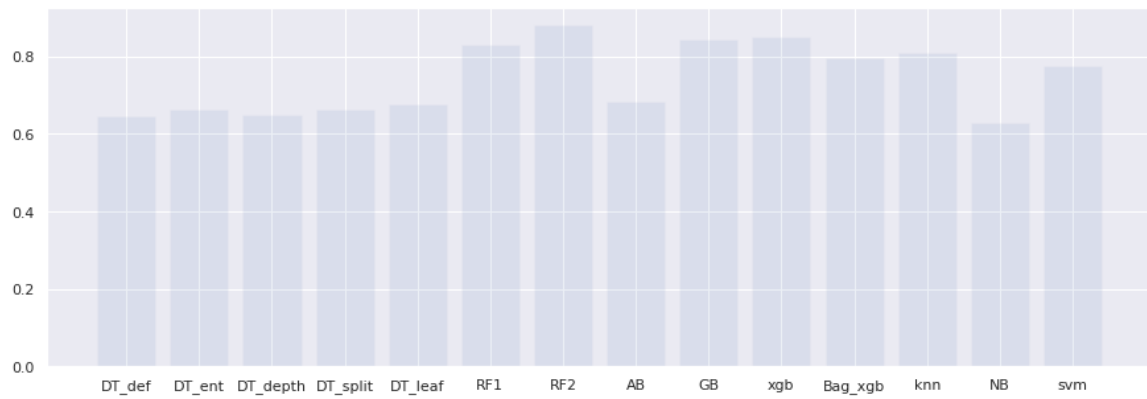
Comparison of test_recall_weighted feature of classification algorithms



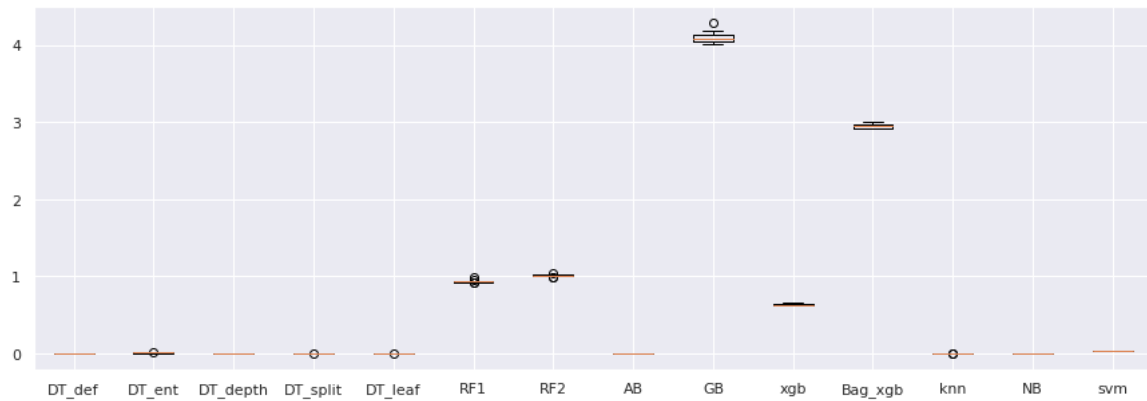
Comparison of test_f1_weighted feature of classification algorithms



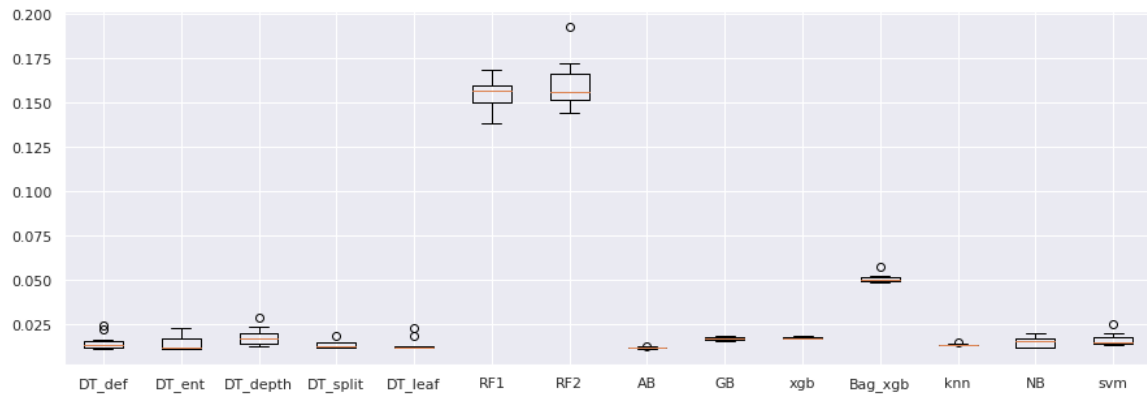
Comparison of test_roc_auc_ovo_weighted feature of classification algorithms



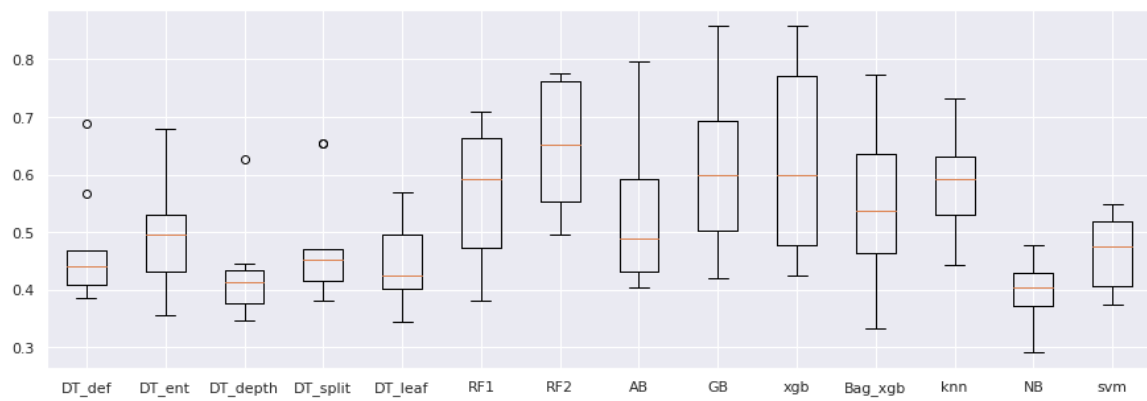
Comparison of score fit_time of classification algorithms



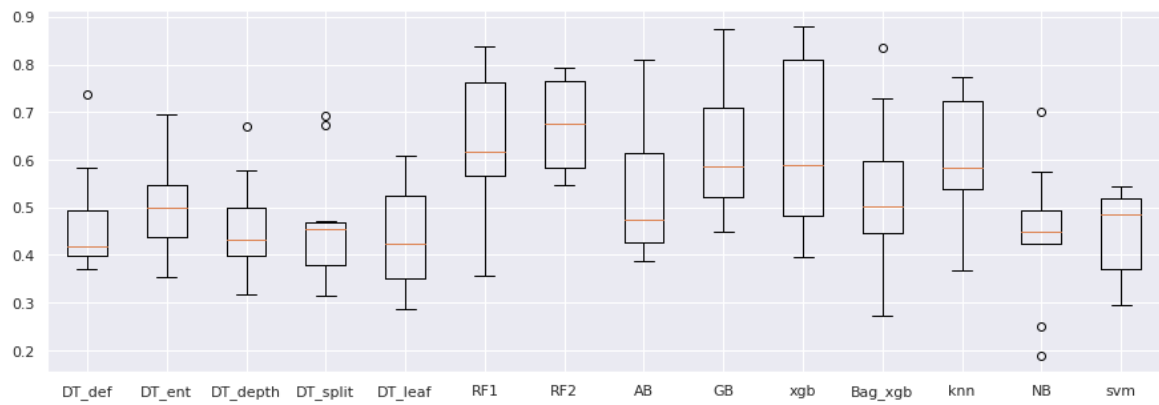
Comparison of score score_time of classification algorithms



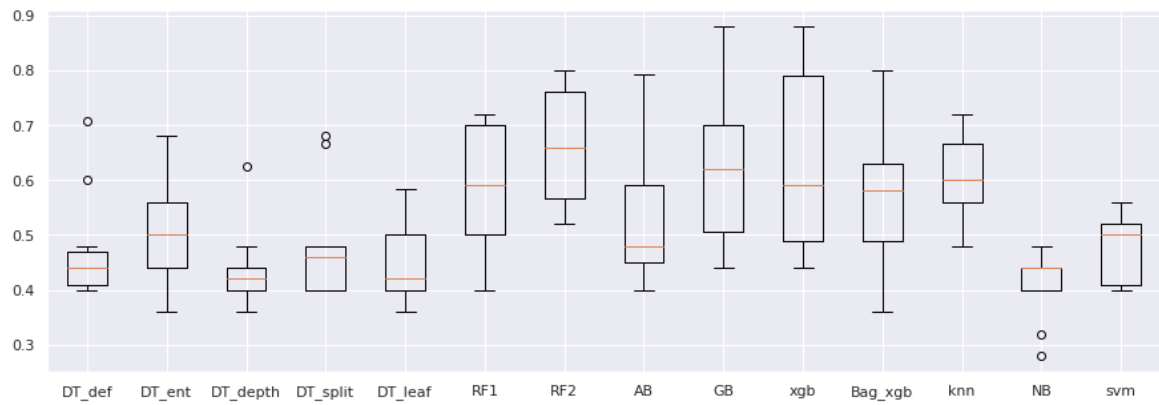
Comparison of score test_balanced_accuracy of classification algorithms



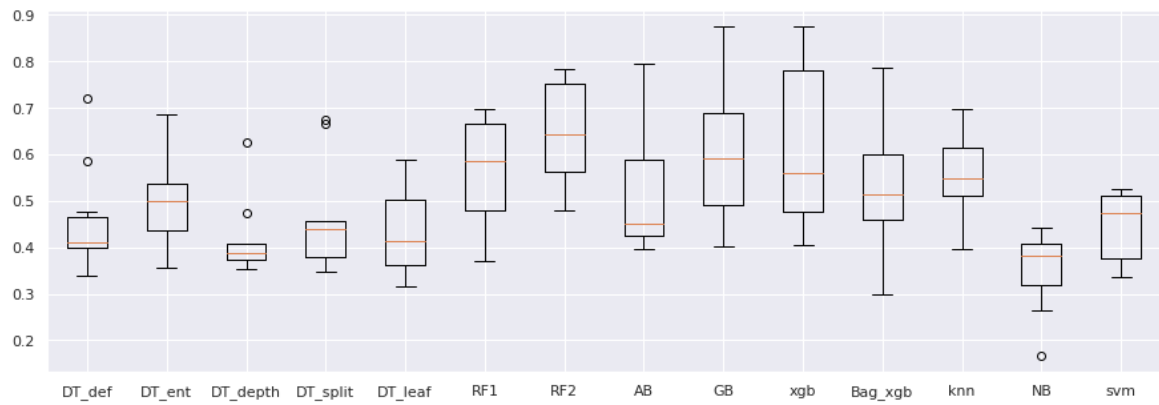
Comparison of score test_precision_weighted of classification algorithms



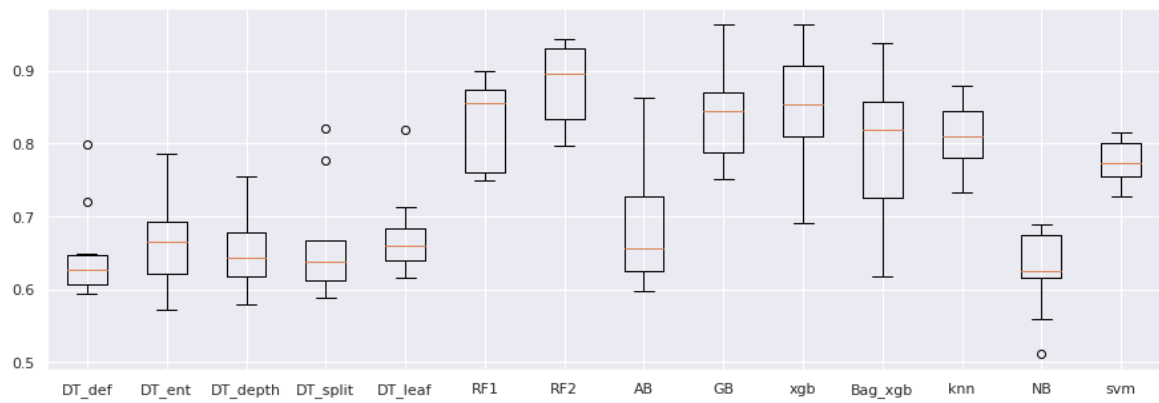
Comparison of score test_recall_weighted of classification algorithms



Comparison of score test_f1_weighted of classification algorithms



Comparison of score test_roc_auc_ovo_weighted of classification algorithms



V prípade keď cieľový atribút rozdelím do 4 skupín je ľahko vidieť, že najlepšie výsledky sa podarilo dosiahnuť pomocou metódy RandomForest, kde precision bol na úrovni 67%, avšak má najdlhší čas tréovania. Druhé najlepšie výsledky sa podarilo dosiahnuť pomocou ExtremeGradientBoostingu a to precision na úrovni 63% a tretí najlepší výsledok sa podarilo získať pomocou GradientBoostingu a to precision na úrovni 62%. Potvrdilo sa aj očakávanie, že najhoršie výsledky dosiahne práve NaiveBayes s precision 44%, čo je ale lepšie ako v prípade rozdelenia cieľového atribútu do 3 skupín.

Záver

Úlohou projektu bolo zostaviť klasifikátor, ktorý na základe údajov o výžive dokáže klasifikovať či sa osoba s ochorením covid-19 z ochorenia vylieči. Pričom zadanie požadovalo použitie algoritmov strojového učenia. Cieľový atribút bol vytvorený z atribútov recovered a confirmed a následne boli atribúty recovered, active, deaths, confirmed odstránené.

- V datasete niektoré informácie chýbali alebo potrebné ich doplniť.
- Dataset bol nevyvážený a bol potrebný oversampling (priniesol kľúčové zlepšenie), na ktorý boli použité algoritmy ADASYN a MDO, pričom sa ukázalo, že na tomto datasete dosahuje mierne lepšie výsledky algoritmus ADASYN. A to v prípade najlepšieho algoritmu (RandomForest-precision ADASYN 86% vs MDO 82%) bolo v precision rozdiel 4%. Táto situácia bola pozorovaná v prípade, keď sa cieľový atribút rozdelil na 2 skupiny. V prípadoch keď som rozdelil cieľovú premennú do 3 a 4 skupín používam algoritmus ADASYN.
- Testoval som nasledujúce algoritmy aj s rôznymi parametrami:
DecisionTreeClassifier, RandomForestClassifier,
AdaBoostClassifier, GradientBoostingClassifier, XGBClassifier,
BaggingClassifier(XGBClassifier), KNeighborsClassifier,
GaussianNB, SVC
- Najlepší precision bol dosiahnutý vo všetkých prípadoch algoritmom RandomForest a to pre oversamplovací algoritmus ADASYN keď cieľový atribút bol rozdelený na 2, 3, 4 skupiny boli dosiahnuté výsledky v poradí 86%, 78%, 67%. Takže sa potvrdilo očakávanie, že s rastúcim počtom skupín bude precision klesať.
- Najhorší precision bol dosiahnutý vo všetkých prípadoch algoritmom NaiveBayes a to pre oversamplovací algoritmus ADASYN keď cieľový atribút bol rozdelený na 2, 3, 4 skupiny boli dosiahnuté výsledky v poradí 68%, 38%, 44%. Zaujímavé je, že pri zvýšení počtu skupín cieľového atribútu z 3 na 4 došlo k zlepšeniu.

Na základe týchto meraní je možné tvrdiť, že sa mi podarilo zostrojiť klasifikátor, ktorý na základe údajov o výžive dokáže s dostatočne vysokou pravdepodobnosťou klasifikovať či sa osoba s ochorením covid-19 z ochorenia vylieči, pričom skupina 0 reprezentuje najmenšiu pravdepodobnosť vyliečenia.