Anàlisis de resultats

Cas Kaggle

Marc Rodriguez Cañero

1527739

Índex

1. Introducció

En aquest document analitzarem els resultats i explicarem tot el que hem fet en el cas Kaggle. Aquest document tracta sobre el notebook realitzat, per tant, és recomendable haver-lo vist abans de llegir el document.

La base de dades assignada en el meu cas és la següent:
https://www.kaggle.com/datasets/bobbyscience/league-of-legends-diamond-ranked-games-10-min

Traca sobre el joc League of Legends. League of Legends és un MOBA (camp de batalla en línia multijugador) on s'enfronten 2 equips (blau i vermell). Hi ha 3 carrils, una jungla i 5 rols. L'objectiu és derrotar el Nexus enemic per guanyar el joc.

2. Anàlisis base de dades

Abans de començar a interactuar amb la base de dades fent servir diferents models, cal realitzar un anàlisis. El que fem en aquest apartat és analitzar els diferents atributs que la composen, per poder entendre'ls, adaptar la nostre base de dades per a fer servir correctament els models i es fixarà quin és l'atribut objectiu a predir de tots els que hi ha a la base de dades justificant el perquè de la decisió.

2.1. Explicació base de dades

La base de dades està formada per 9879 files i 4 columnes.

Cada fila mostrarà les estadístiques de una partida disputada en aquest joc (fins a un total de 9879 jocs ja que té 9879 files). En la taula podem veure les 5 primeres files de la base de dades. Les columnes seran diferents atributs o característiques del joc. Com he comentat anteriorment, el joc està format per dos equips, un blau i un altre vermell, per això, podem veure que alguns atributs tenen la paraula red o blue, per especificar a quin equip pertany el atribut.

Dimensionalitat de la BBDD: (9879, 40)

Tabla de la BBDD:

	gameId	blueWins	blueWardsPlaced	blueWardsDestroyed	blueFirstBlood	blueKills	blueDeaths	blueAssists	blueEliteMonsters	blueDragons	...	redTower
0	4519157822	0	28	2	1	9	6	11	0	0	...	
1	4523371949	0	12	1	0	5	5	5	0	0	...	
2	4521474530	0	15	0	0	7	11	4	1	1	...	
3	4524384067	0	43	1	0	4	5	5	1	0	...	
4	4436033771	0	75	4	0	6	6	6	0	0	...	

5 rows × 40 columns

Figura 1. Base de dades original (inicialment)

2.2. Adaptar base de dades

Al analitzar la base de dades, ens podem adonar que podem tenir atributs que no tenen el tipus correcte, atributs que no necessitem perquè no ens aporten informació, valors que no ens ajudaran a la hora de realitzar els models…

Per això cal fer una adaptació (canvis) en la base de dades, dit d'una altre manera, cal preparar-la correctament per a posteriorment utilitzar els diferents models.

El que hem fet és, una neteja de la base de dades, canviant el tipus (float, int…) de tots els atributs, hem eliminat els atributs (columnes) que hem considerat necessaris (atribut gameID) i hem normalitzat la base de dades.

D'aquesta manera, hem creat dos noves bases de dades. Amb aquestes base de dades es realitzaran prediccions i les anomenarem **base de dades original,** que serà la base de dades original, després de haver-li realitzat la neteja i eliminat els atributs considerats i **la base de dades normalitzada**, que serà la mateixa que la base de dades original, amb la única diferencia de que les dades estan normalitzades.

Dimensionalitat de la BBDD: (9879, 39)

Tabla de la BBDD:

	blueWins	blueWardsPlaced	blueWardsDestroyed	blueFirstBlood	blueKills	blueDeaths	blueAssists	blueEliteMonsters	blueDragons	blueHeralds	...	redTowe
0	-0.998	0.317	-0.379	0.990	0.935	-0.047	1.071	-0.879	-0.753	-0.481	...	
1	-0.998	-0.571	-0.839	-1.010	-0.393	-0.388	-0.405	-0.879	-0.753	-0.481	...	
2	-0.998	-0.404	-1.299	-1.010	0.271	1.657	-0.651	0.719	1.328	-0.481	...	
3	-0.998	1.149	-0.839	-1.010	-0.725	-0.388	-0.405	0.719	-0.753	2.078	...	
4	-0.998	2.925	0.540	-1.010	-0.061	-0.047	-0.159	-0.879	-0.753	-0.481	...	

5 rows × 39 columns

Figura 2. Base de dades normalitzada

Dimensionalitat de la BBDD: (9879, 39)

Tabla de la BBDD:

	blueWins	blueWardsPlaced	blueWardsDestroyed	blueFirstBlood	blueKills	blueDeaths	blueAssists	blueEliteMonsters	blueDragons	blueHeralds	...	redTowe
0	0.000	28.000	2.000	1.000	9.000	6.000	11.000	0.000	0.000	0.000	...	
1	0.000	12.000	1.000	0.000	5.000	5.000	5.000	0.000	0.000	0.000	...	
2	0.000	15.000	0.000	0.000	7.000	11.000	4.000	1.000	1.000	0.000	...	
3	0.000	43.000	1.000	0.000	4.000	5.000	5.000	1.000	0.000	1.000	...	
4	0.000	75.000	4.000	0.000	6.000	6.000	6.000	0.000	0.000	0.000	...	

5 rows × 39 columns

Figura 3. Base de dades original

La nostra variable objectiu serà l'atribut blueWins.

Aquest és el més important de tots perquè és un atribut binari que indica quin equip guanya la partida. Tindrà el valor de 1 si el equip blau guanya la partida o un atribut de 0 si el equip vermell guanya.

Per a realitzar les nostres prediccions, al ser una base de dades que tracta sobre un joc, l'atribut que més ens interessa saber és el que ens indica el guanyador, ja que a tots els jocs es juga amb la intenció de guanyar.

3. Preprocessament

El processament de les dades tracta en reduir la dimensionalitat segons diferents patrons i correlacions.

Quan es treballa en dades n-dimensionals (més d'un atribut), una opció és reduir la seva n-dimensionalitat i escollir 2,3,4... components principals, obtenint unes dades que (ara sí) poden ser visualitzables en un nou espai. Per això, aplicarem dues técniques, PCA (Anàlisis Principal de les Components) i TNSE (t-nse).

3.1. PCA

Com que en aquest punt tenim 39 columnes (ja que em eliminat una), podem escollir multiples opcions a la hora d'escollir el número de components prinicipals quan fem PCA. El que farem es que realitzarem PCA escollint 2,3 i 4 components principals i analitzarem la seva variança en cada cas. D'aquesta manera podrem saber el número de component principals òptim.

A la hora de fer els càlculs de la variança acumulada, a mesura que augmentem el número de components principals augmenta la variança. Per tant, el número de components principals més òptim és de 4 (número més elevat).

Més endavant, a la hora de fer models, quan es vulgui fer proves aplicant la técnica PCA, s'utilitzarà PCA amb 4 components principals.

Dimensionalitat de la BBDD: (9879, 4)

Tabla de la BBDD:

	0	1	2	3
0	-0.588	2.399	-0.061	1.834
1	4.217	-0.232	1.848	2.081
2	2.303	3.253	-2.452	2.154
3	1.388	-0.986	-1.679	0.129
4	1.340	-0.295	1.659	-0.905

Figura 4. Base de dades PCA

3.2. TSNE

De la mateixa manera que hem fet amb PCA, realitzarem TNSE escollint 2 i 3 components principals (no podem escollir 4 ja que tsne ens diu que el nombre de components principals ha de ser inferior a 4) i analitzarem la seva variança en cada cas. D'aquesta manera podrem saber el número de component principals òptim.

A la hora de fer els càlculs de la variança acumulada, a mesura que augmentem el número de components principals disminueix la variança. Per tant, el número de components principals més òptim és de 2 (número més petit).

Més endavant, a la hora de fer models, quan es vulgui fer proves aplicant la técnica TSNE, s'utilitzarà TSNE amb 2 components principals.

Dimensionalitat de la BBDD: (9879, 2)

Tabla de la BBDD:

	0	1
0	-31.314	-14.353
1	80.015	12.161
2	47.848	-10.894
3	11.330	9.458
4	13.260	46.979

Figura 5. Base de dades TSNE

4. Models bàsics

Els primers models que crearem són els models bàsics de scikit-learn, que hem considerat que seran svm (amb kernel gaussiana), regressió lineal i regressió logística.

4.1. Primers models

El que cambiaran en les diferents prediccions serà, el conjunt de test i train.

	40% test i 60% train	**50% test i 50% train**	**60% test i 40% train**
SVM	Precisió: 0.7214 MSE: 0.2785 Temps: 3.3684	Precisió: 0.7255 MSE: 0.2744 Temps: 4.2169	Precisió: 0.7239 MSE: 0.2720 Temps: 2.1056
Regressió lineal	Precisió: 0.2681 MSE: 0.1829 Temps: 1.0087	Precisió: 0.2694 MSE: 0.1826 Temps: 0.0081	Precisió: 0.2742 MSE: 0.1814 Temps: 0.0129
Regressió logística	Precisió: 0.7216 MSE: 0.2783 Temps: 0.0671	Precisió: 0.7246 MSE: 0.2753 Temps: 0.0659	Precisió: 0.7262 MSE: 0.2737 Temps: 0.0598

Taula 1. Models per decidir conjunts d'aprenentatge

L'objectiu d'aquest primer anàlisis, és obtenir els millors valors dels conjunts d'aprenentatge (train-test). Inicialment, només tenim dos conjunts d'aprenentatge, però més endavant, en altres models, en tindrem tres (train-val-test). Per aquest motiu també, inicialment només hem realitzat les prediccions utilitzant la **base de dades original**.

En aquest primers anàlisis, podem extreure les primeres conclusions de que la precisió varia molt poc a pesar de que variem el percentatge de valors que s'agafen per el conjunt de test i train. Tot i així, podem veure que quan obtenim millors resultats, de precisió i temps és agafant el 60% de les dades per a test i el 40% per a train. Per tant, en els anàlisis posteriors utilitzarem aquests percentatges, és a dir, aquesta distribució dels conjunts de train i test.

També podem arribar a la conclusió de que, de moment, svm amb kernel gaussiana és el model que millor precisió té, tot i que la diferència amb regressió logística és mínima i, en canvi, svm triga molt 2 segons més (aproximadament). **Per tant, la regressió logística seria el millor model amb una precisió del 72,62% i un temps de 0.0598 segons.**

4.1. Segons models

Continuant amb els models bàsics mencionats, ara que ja tenim una distribució del conjunt d'aprenentatge, el que farem és comparar els resultats dels models utilitzant les diferents bases de dades que tenim: la base de dades normalitzada, la base de dades amb PCA i la base de dades amb TSNE.

	Base de dades normalitzada	**Base de dades TSNE**	**Base de dades PCA**	**Base de dades original**
SVM	Precisió: 0.7240 MSE: 0.2759 Temps: 3.6854	Precisió: 0.6403 MSE: 0.3596 Temps: 6.0856	Precisió: 0.5561 MSE: 0.4438 Temps: 2.4517	Precisió: 0.7279 MSE: 0.2720 Temps: 2.1056
Regressió lineal	Precisió: 0.2742 MSE: 0.1814 Temps: 0.0089	Precisió: 0.1201 MSE: 0.2199 Temps: 0.5436	Precisió: 0.0030 MSE: 0.2492 Temps: 0.0029	Precisió: 0.2742 MSE: 0.1814 Temps: 0.0129
Regressió logística	Precisió: 0.7289 MSE: 0.2720 Temps: 0.0229	Precisió: 0.6411 MSE: 0.3588 Temps: 0.088	Precisió: 0.5507 MSE: 0.4492 Temps: 0.069	Precisió: 0.7262 MSE: 0.2737 Temps: 0.0598

Taula 2. Models

Utilitzant altres bases de dades, podem que la que millor resultat de precisió obtenim és **regressió logística (60 test i 40 train) amb la base de dades normalitzada amb una precisió de 72,62 % i un temps de 0.0598,** superant a la regressió logística amb la base de dades original que era el millor fins ara. En la base de dades amb TSNE el resultats són una mica pitjors i el que més ens sorprèn és que la base de dades amb PCA és amb la que pitjor resultats obtenim.

En comparació amb el primer anàlisis, podem veure que tot i que els resultats de precisió són similars en els millors resultats, el temps en aquestes base de dades és menor que en la base de dades original. Ho hem aconseguit gràcies a les tècniques de preprocessament aplicades.

Per analitzar els resultats posteriors, el que farem és anar actualitzant aquesta taula poder comparar els models millor visualtment i anar veient els canvis, ja que, com hem dit, ja tenim definits els conjunts d'aprenentatge.

4. Models avançats

En aquest apartat probarem a realitzar prediccions amb models més avançats, o menys coneguts. Els models que utilitzarem seran K-nearest Neighbors (KNN) i Random Forest.

D'acord amb els passos que hem seguit en el anterior apartat per als diferents models, el que farem és cambiar només les bases de dades (original, normalitzada, PCA i TSNE) i utilitzarem el mateix conjunt d'aprenentatge (60% test i 40% train). D'aquesta manera, al tenir els mateixos valors, podrem comparar aquests models amb els anteriors.

	Base de dades normalitzada	**Base de dades TSNE**	**Base de dades PCA**	**Base de dades original**
SVM	Precisió: 0.7240 MSE: 0.2759 Temps: 3.6854	Precisió: 0.6403 MSE: 0.3596 Temps: 6.0856	Precisió: 0.5561 MSE: 0.4438 Temps: 2.4517	Precisió: 0.7279 MSE: 0.2720 Temps: 2.1056
Regressió lineal	Precisió: 0.2742 MSE: 0.1814 Temps: 0.0089	Precisió: 0.1201 MSE: 0.2199 Temps: 0.5436	Precisió: 0.0030 MSE: 0.2492 Temps: 0.0029	Precisió: 0.2742 MSE: 0.1814 Temps: 0.0129
Regressió logística	Precisió: 0.7289 MSE: 0.2720 Temps: 0.0229	Precisió: 0.6411 MSE: 0.3588 Temps: 0.088	Precisió: 0.5507 MSE: 0.4492 Temps: 0.069	Precisió: 0.7262 MSE: 0.2737 Temps: 0.0598
KNN	Precisió: 0.6914 MSE: 0.3085 Temps: 0.6634	Precisió: 0.6379 MSE: 0.3620 Temps: 0.1471	Precisió: 0.5185 MSE: 0.4814 Temps: 0.334	Precisió: 0.6833 MSE: 0.3166 Temps: 1.6248
Random Forest	Precisió: 0.7162 MSE: 0.2837 Temps: 0.9612	Precisió: 0.5927 MSE: 0.4072 Temps: 0.4820	Precisió: 0.5212 MSE: 0.4787 Temps: 0.5738	Precisió: 0.703 MSE: 0.2796 Temps: 1.9164

Taula 2. Models

Entre els dos nous models amb el que obtenim millors resultats de precisió és amb Random Forest, tot i que també tarda una mica més de temps (diferència molt petita).

Amb comparació amb els models bàsics, podem veure que les prediccions no són millor, per tant, aquests dos nous models més avançats no ens aporten gaire si el nostre objectiu és obtenir millors resultats en la predicció.

4. K-FOLD (CrossValidation)

Anteriorment, ja hem parlat de la distribució dels conjunts d'aprenentatge i ja hem mencionat la existencia de tres conjunts, train-val-test, tot i que fins ara només hem distribuit les dades en dos conjunts (train-test).

Perquè afegim la existencia de un nou conjunt de validació (val)?

Al utilitzar diferents models, com per exemple SMV, es pot produir una situació d'overfitting en el conjunt de prova degut a que els paràmetres es poden modificar fins que el estimador funcioni de forma óptima. Per resoldre aquest problema podem crear un altre conjunt de dades, el conjunt de validació. El entrenament continúa en el conjunt d'entrenament, però després és realitza la evaluació en el conjunt de validació, y quan té éxito, la validación continua en el conjunt de prova.

De totes formes, al dividir les dades en tres conjunts, reduim la quantitat de mostres que es poden utilitzar per aprendre el model y els resultat poden dependre de eleccions aleatòrias per el parell de conjunt (entrenament, validació).

Una solució a aquest problema és el procediment validació creuada utilitzant iteradors. Aquesta tècnica es basa en dividir el conjunt d'entrenament en diferents subconjunts. Per decidir en quants subconjunts és dividirá, utilitzarem la funció K-fold, a la qual li passarem un paràmetre K (n-splits) que especificarà en quants subconjunts es divideix el conjunt de dades d'entrenament.

	Base de dades normalitzada	**Base de dades TSNE**	**Base de dades PCA**	**Base de dades original**
SVM	Precisió: 0.7240 MSE: 0.2759 Temps: 3.6854	Precisió: 0.6403 MSE: 0.3596 Temps: 6.0856	Precisió: 0.5561 MSE: 0.4438 Temps: 2.4517	Precisió: 0.7279 MSE: 0.2720 Temps: 2.1056
Regressió lineal	Precisió: 0.2742 MSE: 0.1814 Temps: 0.0089	Precisió: 0.1201 MSE: 0.2199 Temps: 0.5436	Precisió: 0.0030 MSE: 0.2492 Temps: 0.0029	Precisió: 0.2742 MSE: 0.1814 Temps: 0.0129
Regressió logística	Precisió: 0.7289 MSE: 0.2720 Temps: 0.0229	Precisió: 0.6411 MSE: 0.3588 Temps: 0.088	Precisió: 0.5507 MSE: 0.4492 Temps: 0.069	Precisió: 0.7262 MSE: 0.2737 Temps: 0.0598
KNN	Precisió: 0.6914 MSE: 0.3085 Temps: 0.6634	Precisió: 0.6379 MSE: 0.3620 Temps: 0.1471	Precisió: 0.5185 MSE: 0.4814 Temps: 0.334	Precisió: 0.6833 MSE: 0.3166 Temps: 1.6248
Random Forest	Precisió: 0.7162 MSE: 0.2837 Temps: 0.9612	Precisió: 0.5927 MSE: 0.4072 Temps: 0.4820	Precisió: 0.5212 MSE: 0.4787 Temps: 0.5738	Precisió: 0.703 MSE: 0.2796 Temps: 1.9164
K-Fold				
K = 2	Precisió: 0.7382 MSE: 0.2617 Temps: 0.2097	…	…	..
K=3	Precisió: 0.7333 MSE: 0.2662 Temps: 0.2178	…	…	..
K=4	Precisió: 0.7412 MSE: 0.2587 Temps: 0.2686	…	..	..
K=5	Precisió: 0.7494 MSE: 0.2505 Temps: 0.3059	..	…	..

K=6	Precisió: 0.7381 MSE: 0.2618 Temps: 0.3575	…	…	…

Taula 3. Models i K-fold

Com podem veure tenim caselles buides, això és degut a que només em utilitzat K-fold amb el model que millors resultats ens ha donat fins ara, que és el model de regressió logística (amb 60% test) amb la base de dades normalitzada. El conjunt train i val depenen del valors de K (n-splits).

Amb la K que obtenim millors resultats és amb la K igual a 2 (ens quedem amb la iteració que té millor precisió), és a dir, en dividir el conjunt en 60% test, 20% entrenament i 20% de validació. A més a més, no hi ha gaire diferència entre aquesta K i les altres.

Comparant amb els altres models, al utilitzar K-fold és amb el que obtenim millor resultats de precisió, per tant, **el model que millor funciona ara serà K-fold amb una nova distribució del conjunt d'aprenentatge amb 60% test, 20% train i 20% val, que ens donarà una precisió del 74,94% i un temps de 0.3059,** superant a la regressió logística amb la base de dades normalitzada (60% test i 20% entrenament) que era el millor fins ara**.**

5. Hiperparàmetres

Els hiperparàmetres són paràmetres que no s'aprenen directament dins dels estimadors i son els que s'utilitzen per a entrenar el model. A scikit-learn, es passen com a arguments al constructor de les classes de l'estimador. Per exemple, el paràmetre C, kernel, tol...

En els anàlisis anteriors, no hem utilitzar gairebé paràmetres sinó que hem fet servir els paràmetres per defecte de cada funció. En aquest apartat el que farem és identificar els millors paràmetres per a cada model i veurem si al utilitzar aquests paràmetres obtenim millores respecte els anteriors anàlisis.

Per identificar els millors paràmetres tenim el mètodes GridSearchCV, considera exhaustivament totes les combinacions de paràmetres, i RandomizedSearchCV, que pot mostrar un nombre determinat de candidats d'un espai de paràmetres amb una distribució específica.

Degut a que GridSearchCV i RandomizedSearchCV també utilitzen validació creuada (K-fold) i com en el apartat anterior, la K amb millors resultats k = 4, és la que utilitzarem en aquest apartat. En comptes de n-splits, es defineix la k com a cv (passat com a paràmetre GridSearchCV i RandomizedSearchCV).

El model que utilitzarem serà els que millors resultats (sense comptar K-fold ja que també s'aplicarà) han tingut fins ara que és el model de regressió logística i SVM (amb 60% test) amb la base de dades normalitzada.

	Base de dades normalitzada	**Base de dades TSNE**	**Base de dades PCA**	**Base de dades original**
SVM	Precisió: 0.7240 MSE: 0.2759 Temps: 3.6854	Precisió: 0.6403 MSE: 0.3596 Temps: 6.0856	Precisió: 0.5561 MSE: 0.4438 Temps: 2.4517	Precisió: 0.7279 MSE: 0.2720 Temps: 2.1056
Regressió lineal	Precisió: 0.2742 MSE: 0.1814 Temps: 0.0089	Precisió: 0.1201 MSE: 0.2199 Temps: 0.5436	Precisió: 0.0030 MSE: 0.2492 Temps: 0.0029	Precisió: 0.2742 MSE: 0.1814 Temps: 0.0129
Regressió logística	Precisió: 0.7289 MSE: 0.2720 Temps: 0.0229	Precisió: 0.6411 MSE: 0.3588 Temps: 0.088	Precisió: 0.5507 MSE: 0.4492 Temps: 0.069	Precisió: 0.7262 MSE: 0.2737 Temps: 0.0598
KNN	Precisió: 0.6914 MSE: 0.3085 Temps: 0.6634	Precisió: 0.6379 MSE: 0.3620 Temps: 0.1471	Precisió: 0.5185 MSE: 0.4814 Temps: 0.334	Precisió: 0.6833 MSE: 0.3166 Temps: 1.6248
Random Forest	Precisió: 0.7162 MSE: 0.2837 Temps: 0.9612	Precisió: 0.5927 MSE: 0.4072 Temps: 0.4820	Precisió: 0.5212 MSE: 0.4787 Temps: 0.5738	Precisió: 0.703 MSE: 0.2796 Temps: 1.9164
K-Fold				
K = 2	Precisió: 0.7382 MSE: 0.2617 Temps: 0.2097	…	…	..
K=3	Precisió: 0.7333 MSE: 0.2662 Temps: 0.2178	…	…	..
K=4	Precisió: 0.7412 MSE: 0.2587 Temps: 0.2686	…	..	..
K=5	Precisió: 0.7494 MSE: 0.2505 Temps: 0.3059	..	…	..
K=6	Precisió: 0.7381 MSE: 0.2618 Temps: 0.3575	…	…	…
GridSearchCV				
Combinació: K-fold = 5 Regressió logística	Precisió: 0.7309 Paràmetres: C: 10	…	…	…

Combinació: **K-fold = 5** **SVM**	Precisió: 0.7286 Paràmetres: kernel: poly C: 0.1	…	…	…
RandomizedSearchCV				
Combinació: **K-fold = 5** **Regressió logística**	Precisió:0.7309 Paràmetres: C: 10	…	…	…
Combinació: **K-fold = 5** **SVM**	Precisió: 0.7286 Paràmetres: kernel: poly C: 0.1	…	…	…

Taula 4. Models, K-fold i hiperparàmetres

De la mateixa manera que en l'anterior apartat, només hem utilitzat els models que millors resultats ens donaven, per això tenim caselles buides.

Al utilitzar GridSearchCV i RandomizedSearchCV, no només podem trobar els millors paràmetres per a obtenir els millors resultats de precisió, sinó que també ens permet combinar dos models.

Els resultats d'aplicar GridSearchCV i RandomizedSearchCV són els mateixos.

Per al model K-fold amb K=5 i SVM, amb els paràmetres C = 0.1 i kernel polinomial (no utilitzada en anteriors anàlisis), es quan obtenim millors resultats.

Per al model K-fold amb K=5 i regressió logística, amb el paràmetre C = 10, és quan obtenim millors resultats.

Dona millors resultats la combinació de regressió logística i K-fold que la combinació de SVM i K-fold.

De totes maneres, cap de les dues combinacions de models és millor que el millor model que teniem fins ara: **K-fold amb el conjunt d'aprenentatge amb 60% test, 20% train i 20% val (K=4), una precisió del 74,94% i un temps de 0.3059.**

5. Anàlisis complet del cas Kaggle

Al llarg del document hem anat detallant els diferents passos que hem aplicat sobre la base de dades i comentant els canvis i els avenços que anàvem observant durant el procés, però en aquest apartat farem un anàlisis complet de tot el que hem realitzat i com ha anat evolucionant el nostre cas.

En primer lloc, ens hem **familiaritzar amb la base de dades**, que ens ha permès analitzar quina seria la nostre variable objectiu durant totes les prediccions. També ens ha permès **adaptar les bases de dades**, creant així quatre base de dades sobre les quals faríem els test dels models, la base de dades original, la base de dades PCA, la base de dades normalitzada i la base de dades TSNE.

Hem continuat, realitzant els primers anàlisis els models bàsics, que ens ha permès detectar amb quin conjunt d'aprenentatge i quina base de dades obtenim millors resultats (prediccions), en quan a temps i precisió. **El millor model seria el model de regressió lineal sobre la base de dades normalitzada amb conjunt d'aprenentatge d'un 60% de test i 40% de train.**

Els anàlisis amb models més avançats ens han donat resultats similars al models bàsics i no ens aportaven cap informació rellevant.

Això cambiaria al arribar a K-fold (validació creuada), quan hem vist que amb **el conjunt d'aprenentatge amb 60% test, 20% train i 20% de val (K=4) sobre la base de dades normalitzada obteniem millors resultats.**

Finalment, hem acabat analitzant utilitzant les tècniques **GridSearchCV i Randomized Search CV**, que ens han permès realitzar combinacions de models, concepte no que no havien contemplat fins arribar a aquest punt, i analitzar quins són els millors paràmetres per als millors models, també afegint un nou concepte, ja que durant tots els anàlisis, pràcticament hem utilitzat els paràmetres per defecte en els models.

Tots els resultats del treball queden resumits en aquest taula final:

	Base de dades normalitzada	**Base de dades TSNE**	**Base de dades PCA**	**Base de dades original**
SVM	Precisió: 0.7240 MSE: 0.2759 Temps: 3.6854	Precisió: 0.6403 MSE: 0.3596 Temps: 6.0856	Precisió: 0.5561 MSE: 0.4438 Temps: 2.4517	Precisió: 0.7279 MSE: 0.2720 Temps: 2.1056
Regressió lineal	Precisió: 0.2742 MSE: 0.1814 Temps: 0.0089	Precisió: 0.1201 MSE: 0.2199 Temps: 0.5436	Precisió: 0.0030 MSE: 0.2492 Temps: 0.0029	Precisió: 0.2742 MSE: 0.1814 Temps: 0.0129
Regressió logística	Precisió: 0.7289 MSE: 0.2720 Temps: 0.0229	Precisió: 0.6411 MSE: 0.3588 Temps: 0.088	Precisió: 0.5507 MSE: 0.4492 Temps: 0.069	Precisió: 0.7262 MSE: 0.2737 Temps: 0.0598
KNN	Precisió: 0.6914 MSE: 0.3085 Temps: 0.6634	Precisió: 0.6379 MSE: 0.3620 Temps: 0.1471	Precisió: 0.5185 MSE: 0.4814 Temps: 0.334	Precisió: 0.6833 MSE: 0.3166 Temps: 1.6248
Random Forest	Precisió: 0.7162 MSE: 0.2837	Precisió: 0.5927 MSE: 0.4072	Precisió: 0.5212 MSE: 0.4787	Precisió: 0.703 MSE: 0.2796

	Temps: 0.9612	Temps: 0.4820	Temps: 0.5738	Temps: 1.9164
K-Fold				
K = 2	Precisió: 0.7382 MSE: 0.2617 Temps: 0.2097	…	…	..
K=3	Precisió: 0.7333 MSE: 0.2662 Temps: 0.2178	…	…	..
K=4	Precisió: 0.7412 MSE: 0.2587 Temps: 0.2686	…	..	..
K=5	Precisió: 0.7494 MSE: 0.2505 Temps: 0.3059	..	…	..
K=6	Precisió: 0.7381 MSE: 0.2618 Temps: 0.3575	…	…	…
GridSearchCV				
Combinació: K-fold = 5 Regressió logística	Precisió:0.7309 Paràmetres: C: 10	…	…	…
Combinació: K-fold = 5 SVM	Precisió: 0.7286 Paràmetres: kernel: poly C: 0.1	…	…	…
RandomizedSearchCV				
Combinació: K-fold = 5 Regressió logística	Precisió:0.7309 Paràmetres: C: 10	…	…	…
Combinació: K-fold = 5 SVM	Precisió: 0.7286 Paràmetres: kernel: poly C: 0.1	…	…	…

Taula 4. Models, K-fold i hiperparàmetres

6. Observacions

Tant en aquests document, com en el notebook he estat utilitzant la primera persona del plural degut a que estem acostumats a realitzar treballs en grup i tinc més facilitat per redactar correctament, a més a més, que al parlar com si fossim un grup, de cara a la persona que ho llegeix, desperta millors sensacions que si utilitzava la primera persona del singular.

7. Annexos

Tot el notebook del treball s'ha guardat i desenvolupat al següent *Github*:

https://github.com/Nara-On/Practica2APC-GrupGPA405-1130