Nițu Andreea

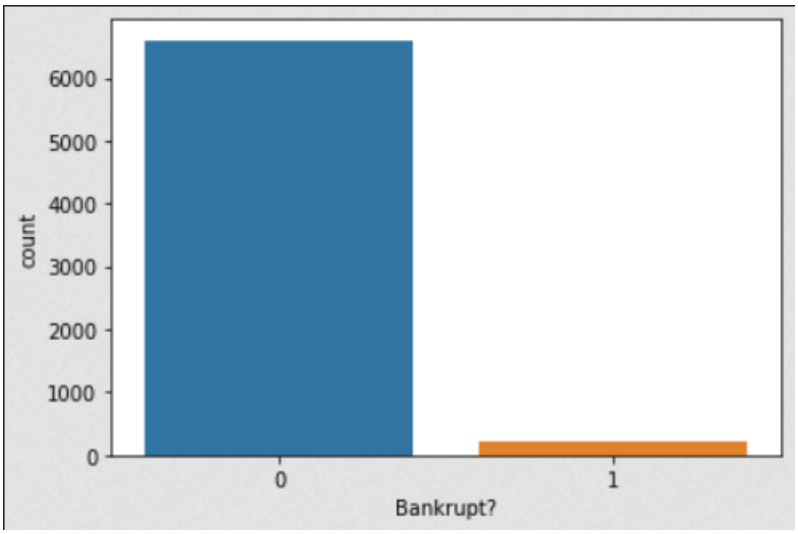
Proiect MAAPO

Setul de date

Am ales setul de date numit Company Bankruptcy Prediction care cuprinde date din Jurnalul economic din Taiwan între anii 1999 și 2009. Setul de date are o dimensiune de 10.9 MB și are 6819 înregistrări cu 95 de atribute fiecare. Setul de date atribuie un label pentru fiecare înregistrare conform căruia știm dacă o companie a falimentat sau nu. Caracteristicile prezente in setul de date reprezinta indicatori economici.

Preprocesare

Se poate observa clar că înregistrările nu sunt balansate. Sunt 220 de companii care au falimentat și restul, 6599, care nu au falimentat.



Distributia datelor

Nicio înregistrare nu are caracteristici lipsa, deci putem sari peste aceasta parte a preprocesarii.
Acuratețea modelelor nu este o metoda buna de a masura calitatea acestora, deci am folosit F1 Score care se calculeaza folosind media armonica dintre precizie și recall.
Datele au fost împărțite după regula 80-10-10 (antrenare, validare, testare).

Înainte de a începe calculele am standardizat datele folosind algoritmul StandardScaler din biblioteca scikit-learn din Python.

$$F1\text{-}score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Algoritmi folositi

Am antrenat următoarele modele și am comparat rezultatele obținute: SVM, Regresie Logistica, Random Forest si Perceptron cu mai multe straturi (Multilayer). Pe langa aceasta, am aplicat 3 metode de reducere a dimensiunii și am analizat efectul acestora asupra calității rezultatelor. Folosind metodele PCA și Truncated SVD am redus dimensiunea la 50 de caracteristici și folosind metoda TSNE am obținut 2 componente.

Rezultate

Pentru a putea determina cel mai bun algoritm, și cea mai buna metoda pentru împărțirea datelor am folosit F1 Score. Am rulat testele după ce am aplicat Standard Scaler.

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score

print(accuracy_score(y_validation.to_numpy().squeeze(), pred))
print(f1_score(y_validation.to_numpy().squeeze(), pred))
```

```
0.9897360703812317
0.9066666666666667
```

F1-Score și acuratețea folosind modelul SVM

```
print(accuracy_score(y_validation.to_numpy().squeeze(), pred))
print(f1_score(y_validation.to_numpy().squeeze(), pred))
```

```
0.9574780058651027
0.4081632653061224
```

F1-Score și acuratețea folosind modelul Regresie Logistica

```
[ ] print(accuracy_score(y_validation.to_numpy().squeeze(), pred))
    print(f1_score(y_validation.to_numpy().squeeze(), pred))

    0.9882697947214076
    0.8888888888888888
```

F1-Score și acuratețea folosind clasificatorul Random Forest

```
[ ] print(accuracy_score(y_validation.to_numpy().squeeze(), pred))
    print(f1_score(y_validation.to_numpy().squeeze(), pred))

    0.9868035190615836
    0.88
```

F1-Score și acuratețea folosind clasificatorul MLP

După ce am aplicat metodele de reducere a dimensiunii pe setul de date, am re-antrenat modelele și am regenerat atat F1-Score cât și matricele de confuzie.
Pentru reducere a dimensiunii am folosit PCA, Truncated SVD si TSNE.

```
[19] pred = svm.predict(X_validation_pca)
     print(accuracy_score(y_validation.to_numpy().squeeze(), pred))
     print(f1_score(y_validation.to_numpy().squeeze(), pred))

     0.906158357771261
     0.0
```

```
svm.fit(X_train_svd, y_train.values.ravel())
pred = svm.predict(X_validation_svd)
print(accuracy_score(y_validation.to_numpy().squeeze(), pred))
print(f1_score(y_validation.to_numpy().squeeze(), pred))

0.9149560117302052
0.0
```

```
svm = SVC(C=10**6)

svm.fit(X_train_tsne, y_train.values.ravel())
pred = svm.predict(X_validation_tsne)
print(accuracy_score(y_validation.to_numpy().squeeze(), pred))
print(f1_score(y_validation.to_numpy().squeeze(), pred))

0.9442815249266863
0.0
```

F1-Score si acuratetea folosind modelul SVM dupa reducerea dimensiunii (PCA, Truncated SVD si TSNE)

```
logreg.fit(X_train_pca, y_train.values.ravel())
pred = logreg.predict(X_validation_pca)

print(accuracy_score(y_validation.to_numpy().squeeze(), pred))
print(f1_score(y_validation.to_numpy().squeeze(), pred))
```

```
0.7536656891495601
0.16
```

```
logreg.fit(X_train_svd, y_train.values.ravel())
pred = logreg.predict(X_validation_svd)

print(accuracy_score(y_validation.to_numpy().squeeze(), pred))
print(f1_score(y_validation.to_numpy().squeeze(), pred))
```

```
0.7903225806451613
0.1437125748502994
```

```
logreg.fit(X_train_tsne, y_train.values.ravel())
pred = logreg.predict(X_validation_tsne)

print(accuracy_score(y_validation.to_numpy().squeeze(), pred))
print(f1_score(y_validation.to_numpy().squeeze(), pred))
```

```
0.9442815249266863
0.0
```

F1-Score si acuratetea folosind Regresia Logistica după reducerea dimensiunii (PCA, Truncated SVD si TSNE)

```
rand_for.fit(X_train_pca, y_train.values.ravel())
pred = rand_for.predict(X_validation_pca)

print(accuracy_score(y_validation.to_numpy().squeeze(), pred))
print(f1_score(y_validation.to_numpy().squeeze(), pred))
```

```
0.9442815249266863
0.0
```

```
rand_for.fit(X_train_svd, y_train.values.ravel())
pred = rand_for.predict(X_validation_svd)

print(accuracy_score(y_validation.to_numpy().squeeze(), pred))
print(f1_score(y_validation.to_numpy().squeeze(), pred))
```

```
0.9442815249266863
0.0
```

```
rand_for.fit(X_train_tsne, y_train.values.ravel())
pred = rand_for.predict(X_validation_tsne)

print(accuracy_score(y_validation.to_numpy().squeeze(), pred))
print(f1_score(y_validation.to_numpy().squeeze(), pred))
```

```
0.9442815249266863
0.0
```

F1-Score și acuratețea folosind clasificatorul Random Forest după reducerea dimensiunii (PCA, Truncated SVD si TSNE)

```
mlp.fit(X_train_pca, y_train.values.ravel())
pred = mlp.predict(X_validation_pca)

print(accuracy_score(y_validation.to_numpy().squeeze(), pred))
print(f1_score(y_validation.to_numpy().squeeze(), pred))
```

```
0.9325513196480938
0.08
```

```
mlp.fit(X_train_svd, y_train.values.ravel())
pred = mlp.predict(X_validation_svd)

print(accuracy_score(y_validation.to_numpy().squeeze(), pred))
print(f1_score(y_validation.to_numpy().squeeze(), pred))
```

```
0.9149560117302052
0.03333333333333333
```

```
mlp.fit(X_train_tsne, y_train.values.ravel())
pred = mlp.predict(X_validation_tsne)

print(accuracy_score(y_validation.to_numpy().squeeze(), pred))
print(f1_score(y_validation.to_numpy().squeeze(), pred))
```

```
0.9442815249266863
0.0
```

F1-Score și acuratețea folosind clasificatorul MLP după reducerea dimensiunii (PCA, Truncated SVD si TSNE)

Deoarece nu se poate folosi acuratețea în acest caz, am generat și matricele de confuzie pentru fiecare model antrenat pentru a putea vedea ce performanta are fiecare. Am decis ca trebuie sa folosesc aceste rezultate alături de F1-Scores pentru a determina care este cel mai performant model.

	0	1
0	641	3
1	4	34

Matricea de confuzie pentru modelul SVM

	0	1
0	643	1
1	28	10

Matricea de confuzie pentru modelul Regresie Logistica

	0	1
0	642	2
1	6	32

Matricea de confuzie pentru clasificatorul Random Forest

	0	1
0	640	4
1	5	33

Matricea de confuzie pentru clasificatorul MLP

La fel cum am facut si pentru F1-Score, am calculat matricele de confuzie si pentru modelele antrenate dupa ce am aplicat metodele de reducere a dimensiunii.

	0	1
0	618	26
1	38	0

	0	1
0	624	20
1	38	0

	0	1
0	644	0
1	38	0

Matricea de confuzie pentru modelul SVM dupa reducerea dimensiunii (PCA, Truncated SVD si TSNE)

	0	1
0	494	150
1	22	16

	0	1
0	527	117
1	26	12

	0	1
0	644	0
1	38	0

Matricea de confuzie pentru modelul Regresie Logistica dupa reducerea dimensiunii (PCA, Truncated SVD si TSNE)

	0	1
0	644	0
1	38	0

	0	1
0	644	0
1	38	0

	0	1
0	644	0
1	38	0

Matricea de confuzie pentru clasificatorul Random Forest dupa reducerea dimensiunii (PCA, Truncated SVD si TSNE)

	0	1
0	636	8
1	36	2

	0	1
0	623	21
1	37	1

	0	1
0	644	0
1	38	0

Matricea de confuzie pentru clasificatorul MLP dupa reducerea dimensiunii (PCA, Truncated SVD si TSNE)

	Dimensiune originala	PCA	Truncated SVD	TSNE
SVM	90.6%	0%	0%	0%
Regresie Logistica	40.8%	16%	14.3%	0%
Random Forest	88.8%	0%	0%	0%
MLP	88%	8%	3.3%	0%

F1-Score-urile pentru fiecare model înainte și după reducerea dimensiunii

Concluzii

Cele mai bune rezultate au fost obținute folosind datele inițiale, fără reducerea dimensiunii. Concluzia la care am ajuns este că după reducerea dimensiunii datelor, modelele nu mai pot invata la fel de bine. Păstrând datele cu dimensiunile originale am obținut cel mai bun F1-Score. Cei patru algoritmi au avut rezultate similare, exceptand Regresia Logistica care a avut cea mai slabă performanță. Cel mai performant model a fost SVM, care a obtinut un F1-Score de 90.6%.