ÍNDEX DE CONTINGUTS

PRÀCTICA 2. INTEGRACIÓ, NETEJA, VALIDACIÓ I ANÀLISI DE LES DADES.

1. DESCRIPCIÓ:

Per aquesta pràctica s'han utilitzat les dades sobre l'enfonsament del Titànic, disponibles a Kaggle. Es poden trobar a l'enllaç: https://www.kaggle.com/c/titanic/data

La pregunta que es vol respondre és: qui va sobreviure i qui va morir en l'enfonsament del Titànic?

Com es veu a la taula següent, es diposa de diverses variables: edad, sexe, classe, número de germans, pares i fills, etc.

Variable	Definition	Key
survived	Survived	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
Age	Age in years	
sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Nota: Els quaderns de Python (ipynb) estan explicats exhaustivament. Aquest document només pretén explicar els passos més importants que s'han realitzat.

2. INTEGRACIÓ, SELECCIÓ I ESTUDI INICIAL DE LES DADES:

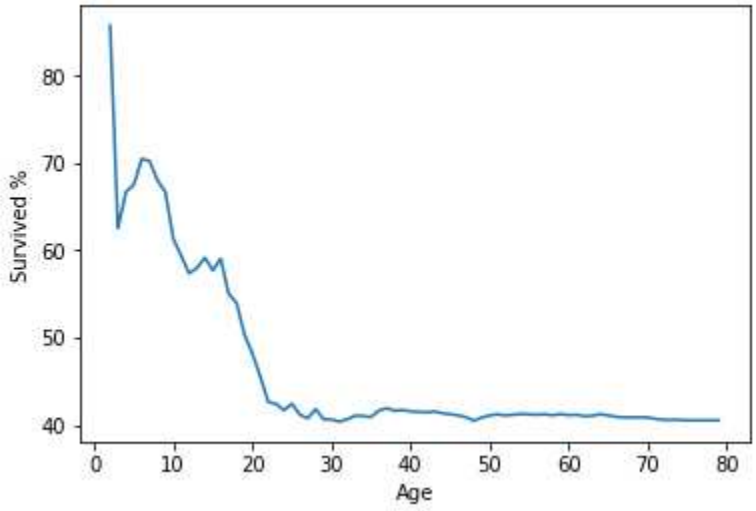
Hi ha 891 dades al conjunt d'entrenament i 418 al de test. Per als anàlisis i procediments s'utilitzaran **únicament** les dades d'entrenament. La idea és mantenir les dades de test ocultes fins al final, evitant qualsevol filtració d'informació.

Així, qui va sobreviure a l'enfonsament? A primer cop d'ull, queda clar que les dones i la gent de classe alta tenien moltes més possibilitats de sobreviure:

```
   Pclass  Survived
0       1  0.629630
1       2  0.472826
2       3  0.242363
```

```
      Sex  Survived
0  female  0.742038
1    male  0.188908
```

Aquest DataSet té pocs atributs numèrics continus. Un d'ells és l'edat. Es diu que els nens i joves van ser evaquats primer, de manera que es van salvar. Anem a comprovar-ho:



Finalment, hi ha dos atributs que semblen complicats d'aprofitar: 'Cabin' i 'Ticket'. Són atributs alfanumèrics, amb una gran quantitat de valors únics (per tant, no són classes).

```
Què ens diuen els dos atributs restants, 'Cabin' i 'Ticket'?
Valors únics per 'Cabin':  147
Valors únics per 'Ticket':  681
```

Per aquest motiu, es decideix descartar-los de l'anàlisi.

Resum inicial:

- L'atribut classe, pel qual es voldria classificar, és 'Survived' i pren els valors 0 i 1.
- S'utilitzaran els atributs 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', i 'Embarked'.

3. Neteja de dades

En aquest apartat s'aplicarà una neteja de les dades. Les tasques a realitzar són:

1. Tractament de valors nuls i extrems.
2. Enginyeria d'atributs. S'extraurà el títol dels noms, creant així un atribut categòric nou.
3. Discretització dels atributs numèrics.

3.1 Valors nuls

- Hi ha valors nuls als atributs 'Age' i 'Embarked' del conjunt d'entrenament.
- Hi ha valors nuls als atributs 'Age' i 'Fare' del conjunt de test.

Un 20% dels valors de 'Age' és nul. Això suposa un problema, perquè com s'ha vist les persones joves tenien més possibilitat de sobreviure. Com que no podem utilitzar valors aleatoris, i una estadística com la mitjana seria injusta, s'ha decidit aplicar **un arbre de regressió** per predir l'edat a partir de les altres variables.

```
Els valors predits són:

[33, 31, 15, 32, 19, 27, 27, 22, 24, 32, 30, 36, 22, 24, 36, 40, 40, 27, 30, 22, 30, 30, 27,
22, 48, 24, 16, 23, 31, 15, 44, 22, 8, 44, 27, 24, 23, 14, 15, 30, 24, 44, 22, 44, 22, 34, 64
5, 27, 27, 26, 47, 32, 19, 19, 36, 13, 22, 28, 27, 21, 24, 27, 19, 34, 25, 32, 30, 44, 26, 19
47, 34, 24, 16, 26, 36, 16, 50, 30, 39, 32, 24, 38, 24, 4, 19, 23, 44, 24, 44, 30, 25, 24, 22
27, 15, 44, 21, 39, 27, 29, 42, 27, 19, 27, 29, 35, 34, 15, 29, 16, 40, 50, 36, 16, 34, 27, 2
44, 16, 56, 39, 19, 29, 44, 24, 30, 58, 8, 33, 24, 16, 26, 27, 13]
[27, 61, 24, 13, 25, 29, 50, 44, 23, 27, 35, 30, 27, 57, 26, 22, 29, 30, 44, 44, 24, 22, 34,
6, 36, 44, 42, 24, 44, 16, 64, 25, 15, 35, 27, 19, 30, 29, 22, 44, 26, 13, 14, 42, 44, 27, 39
30, 28, 24, 24, 23, 22, 42, 32, 24, 16, 40, 29, 44, 43, 26, 44, 47, 31, 16, 22, 30, 30, 36]
```

Per entrenar el model **només s'han utilitzat les dades d'entrenament**. Després s'ha aplicat la transformació en ambdós conjunts (entrenament i test).

Finalment, per els valors nuls de 'Embarked' s'ha fet una propagació cap endavant, simulant un valor aleatori. Per els de 'Fare', s'ha elegit la mediana.

```
Ja no hi ha valors nuls:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 9 columns):
Survived    891 non-null int64
Pclass      891 non-null int64
Name        891 non-null object
Sex         891 non-null object
Age         891 non-null float64
SibSp       891 non-null int64
Parch       891 non-null int64
Fare        891 non-null float64
Embarked    891 non-null object

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 8 columns):
Pclass      418 non-null int64
Name        418 non-null object
Sex         418 non-null object
Age         418 non-null float64
SibSp       418 non-null int64
Parch       418 non-null int64
Fare        418 non-null float64
Embarked    418 non-null object
```

3.2 Valors extrems

A continuació es mostra una descripció estadística bàsica de les dades.

```
Descripció de les variables numèriques:
              Age       SibSp       Parch        Fare
count  891.000000  891.000000  891.000000  891.000000
mean    29.463715    0.523008    0.381594   32.204208
std     13.811493    1.102743    0.806057   49.693429
min      0.420000    0.000000    0.000000    0.000000
25%     21.000000    0.000000    0.000000    7.910400
50%     28.000000    0.000000    0.000000   14.454200
75%     37.000000    1.000000    0.000000   31.000000
max     80.000000    8.000000    6.000000  512.329200
```

Hi ha valors extrems en tots els casos menys 'Age'. La mitjana de 'Fare' és 32.2, mentre que el valor màxim 512.32. Això causa que la desviació sigui també molt elevades. Com que no es vol prendre mesures dràstiques sense més, s'ha decidit **suavitzar** la situació.

- S'ha pres per valor extrem aquell que està més enllà de 3 desviacions estàndards.
- S'ha substituït el seu valor per el valor màxim del conjunt de dades. D'aquesta manera, el valor segueix sent màxim (dins el conjunt), però sense sobresortir.

```
Descripció després de tractar els valors extrems:
               Age       SibSp       Parch        Fare
count  891.000000  891.000000  891.000000  891.000000
mean    29.448002    0.452301    0.345679   29.635367
std     13.760462    0.788059    0.663266   35.532019
min      0.420000    0.000000    0.000000    0.000000
25%     21.000000    0.000000    0.000000    7.910400
50%     28.000000    0.000000    0.000000   14.454200
75%     37.000000    1.000000    0.000000   31.000000
max     70.500000    3.000000    2.000000  164.866700
```

3.3 Enginyeria d'atributs

La idea aquí és:

1. Extreure el títol dels noms.
2. Discretitzar els atributs 'Age' i 'Fare' en dos nous atributs ordinals.

```
#Es fa una altra còpia de seguretat
train_test = [clean_train.copy(), clean_test.copy()]
age_labels = ["(0, 21]", "(21, 28]","(28, 37]","(37, 71]"]
age_bins = [0,21,28,37,71]

fare_labels = ["(0, 8]","(8, 14]","(14, 31]","(31, 165]"]
fare_bins = [-0.5,8,14,31,165]


for tt in train_test:
    tt['L_Age'] = pd.cut(tt['Age'], bins=age_bins, labels=age_labels)
    tt['L_Fare'] = pd.cut(tt['Fare'], bins=fare_bins, labels=fare_labels)
    tt['Title'] = tt["Name"].str.extract(' ([A-Za-z]+)\.', expand=False).replace(['Lady',
    tt['Title'] = tt['Title'].replace('Mlle', 'Miss')
    tt['Title'] = tt['Title'].replace('Ms', 'Miss')
    tt['Title'] = tt['Title'].replace('Mme', 'Mrs')

#Es recupera la el conjunt de test
train_test[1]["PassengerId"] = test_index
print("Qui va sobreviure segons el títol?")
print_survived_by(train_test[0],["Title"])
print("Primeres 5 files de 'Train':\n",train_test[0][:5])
print("\nPrimeres 5 files de 'Test':\n",train_test[1][:5])
```

Amb això s'obté una nova variable d'interès, a la vegada que s'elimina el 'Name'. Llavors, ara podem respondre:

```
Qui va sobreviure segons el títol?
    Title  Survived
3     Mrs  0.793651
1    Miss  0.702703
0  Master  0.575000
4   Other  0.347826
2      Mr  0.156673
```

Després de la neteja, en vermell, els nous atributs:

```
Primeres 5 files de 'Train':
    Survived  Pclass                                               Name  \
0          0       3                            Braund, Mr. Owen Harris
1          1       1  Cumings, Mrs. John Bradley (Florence Briggs Th...
2          1       3                             Heikkinen, Miss. Laina
3          1       1       Futrelle, Mrs. Jacques Heath (Lily May Peel)
4          0       3                           Allen, Mr. William Henry

      Sex   Age  SibSp  Parch     Fare Embarked     L_Age     L_Fare Title
0    male  22.0    1.0    0.0   7.2500        S  (21, 28]      (0, 8]    Mr
1  female  38.0    1.0    0.0  71.2833        C  (37, 71]   (31, 165]   Mrs
2  female  26.0    0.0    0.0   7.9250        S  (21, 28]      (0, 8]  Miss
3  female  35.0    1.0    0.0  53.1000        S  (28, 37]   (31, 165]   Mrs
4    male  35.0    0.0    0.0   8.0500        S  (28, 37]    (8, 14]    Mr
```

S'han generat dos nous fitxers: 'cleaned_train.csv' i 'cleaned_test.csv'.

4. Anàlisi de les dades

Per a l'anàlisi de les dades es partirà dels fitxers creats a l'apartat anterior.

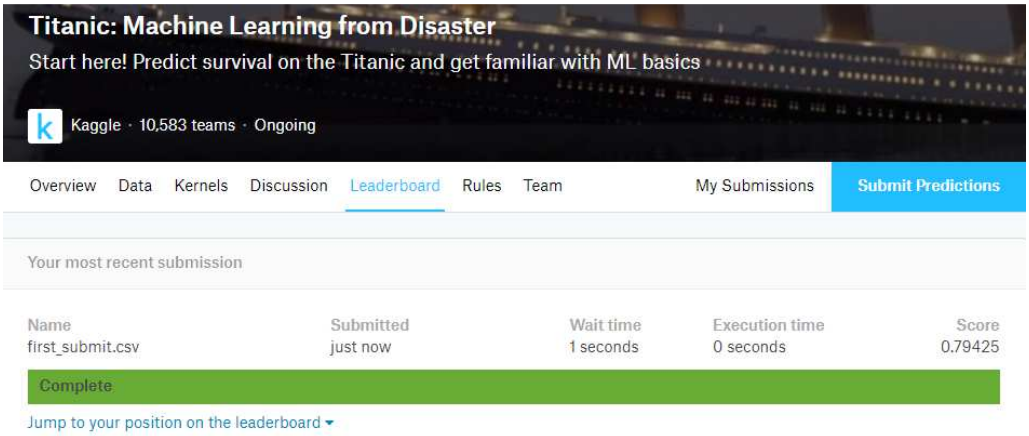
4.1 Selecció dels grups a analitzar i comparar

La idea aquí és decidir quins atributs es faran servir i seleccionar-los. A més, s'aprofitarà per generar un fitxer per presentar a la competició de *Kaggle*.

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import preprocessing

le = preprocessing.LabelEncoder()

y = train["Survived"]
X =train.drop(["Survived","Name","Age","Fare"], axis=1)

X["Title"]= le.fit_transform(X["Title"])
X["Embarked"]= le.fit_transform(X["Embarked"])
X["Sex"]= le.fit_transform(X["Sex"])
X["L_Age"]= le.fit_transform(X["L_Age"])
X["L_Fare"]= le.fit_transform(X["L_Fare"])

X_test = test.drop(["PassengerId","Name","Age","Fare"], axis=1)
X_test["Title"]= le.fit_transform(X_test["Title"])
X_test["Embarked"]= le.fit_transform(X_test["Embarked"])
X_test["Sex"]= le.fit_transform(X_test["Sex"])
X_test["L_Age"]= le.fit_transform(X_test["L_Age"])
X_test["L_Fare"]= le.fit_transform(X_test["L_Fare"])

clf = GradientBoostingClassifier(random_state=0)
clf = clf.fit(X,y)
predict = clf.predict(X_test)

first_submit = pd.DataFrame(test_id.tolist(), columns=["PassengerId"])
first_submit["Survived"] = predict.tolist()
#first_submit.to_csv("first_submit.csv", index=False)
print(X)
```

```
   Pclass  Sex  SibSp  Parch  Embarked  L_Age  L_Fare  Title
0       3    1    1.0    0.0         2      1       0      2
1       1    0    1.0    0.0         0      3       2      3
2       3    0    0.0    0.0         2      1       0      1
3       1    0    1.0    0.0         2      2       2      3
```

S'ha utilitzat el *LabelEncoder* de *Sklearn*. El resultat, com es mostra a la imatge, són un conjunt d'atributs categoritzats en format numèric. Les últimes línies serveixen per generar un fitxer en el format demanat per la competició de *Kaggle*:



- L'algorisme utilitzat és el 'GradientBoosting' de *Sklearn*.
- El resultat obtingut és de 0.79425, força bo per un primer intent (posició 2511 de 10565).

4.2 Comprovació de la normalitat i homogeneïtat de la variància.

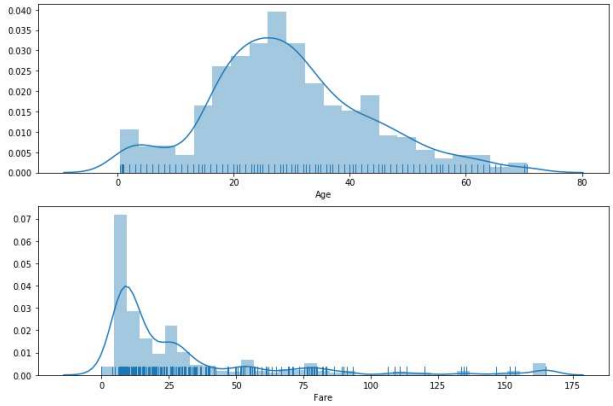
Es disposa d'unes dades principalment categòriques, on no té sentit aplicar test estadístics de normalitat. No obstant, sí que és viable fer-ho per les dues variables continues 'Age' i 'Fare'.

4.2.1 Test basat en D'Agostino i Pearson per la normalitat

La hipòtesi nul·la que sosté aquest test és que les distribucions són normals. Si el p-valor és inferior a 0.05, es rebutjarà la hipòtesis i **no** podrem afirmar que siguin distribucions normals.

```
Age  amb p-value 5.155589746600271e-06
Fare  amb p-value 2.1889771704168276e-92
```

Es rebutja la hipòtesis en ambdós casos. Per comprovar-ho, es dibuixen les distribucions:



La distribució de 'Age' és gairebé normal, mentre que la de 'Fare' no ho és gens, tal i com mostren els p-valors.

4.2.2 Test de Fligner-Killeen per a la homogeneïtat de la variança

Per aquest test, cal agrupar segons la classe. Així, s'agrupa per 'Survived' i es comparen les variàncies d'ambdós grups per als atributs numèrics.

La hipòtesis nul·la d'aquest test és que les variàncies són homogènies.

```
Homogeneitat de la variança de: Age  amb p-value 0.1126575771373211
Homogeneitat de la variança de: Fare  amb p-value 2.466824475528314e-21
                 Age         Fare
Survived
0          177.689239   688.002855
1          205.039282  1898.580840
```

Es rebutja la hipòtesis per 'Fare', però no per 'Age'. A la taula es mostren les variàncies i queda clar que la de 'Fare' és molt diferent. És interessant perquè si es comparen les mitjanes dels grups, el grup supervivent ('Survived' = 1) té un valor molt més alt:

Survived	Age	Fare
0	30.626179	22.117887
1	28.343690	48.395408

És a dir, les persones que havien pagat un 'Fare' més alt, tenien més possibilitats de sobreviure.

4.3 i 5 Correlacions i representació dels resultats

Com s'ha dit, majoritàriament es treballa amb variables categòriques. No té massa sentit seguir aplicant test estadístiques per aquestes dades. Així, s'ha decidit **aprofundir en les correlacions** entre variables.

Per fer-ho, s'ha dissenyat una funció que, donada una llista d'atributs i un atribut per comparar, fa el següent:

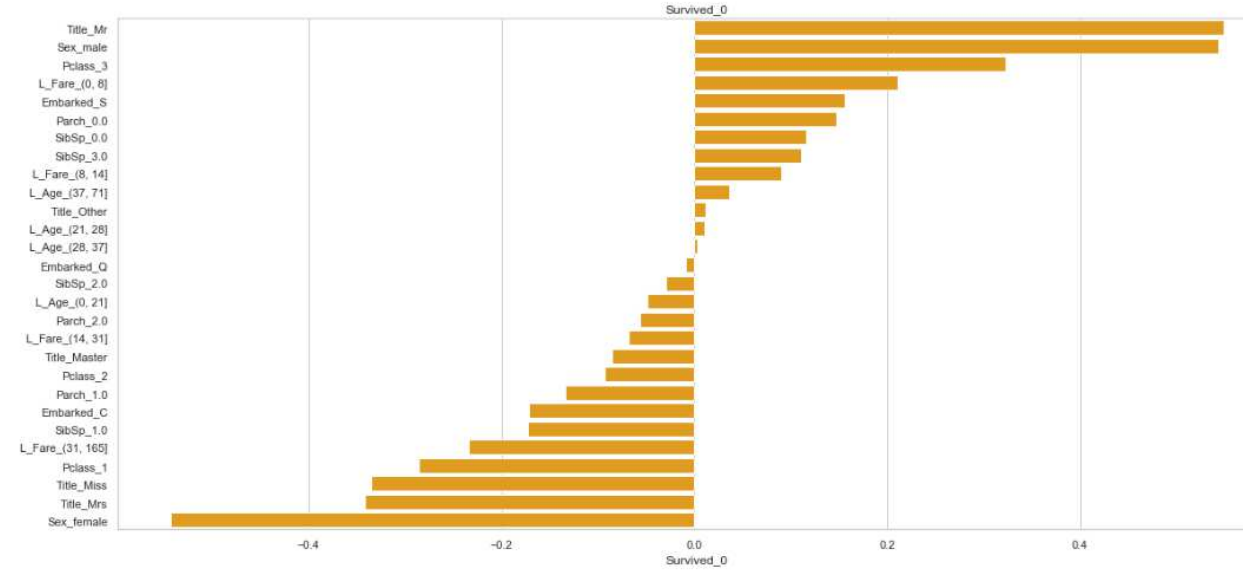
1. Llegeix del fitxer 'cleaned_train.csv' i selecciona els atributs d'interès, igual que s'ha fet abans.
2. Aplica la funció 'Pandas.DataFrame.get_dummies()'. El que fa és, per cada categoria de cada atribut (per exemple, Sex_male i Sex_female), crea una nova columna que el respresenta. Així, una dada només pot tenir actiu (1) una de les noves columnes.

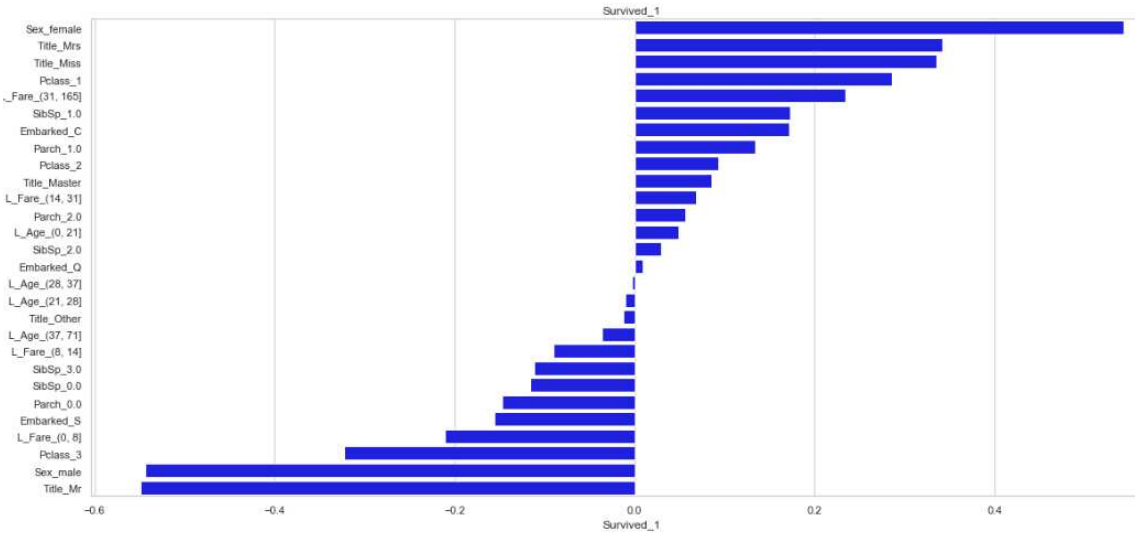
```
   Sex_female  Sex_male
0           0         1
1           1         0
2           1         0
3           1         0
4           0         1
5           0         1
6           0         1
```

3. Calcula **la matriu de correlació** amb la funció 'Pandas.DataFrame.corr()'.
4. Dibuixa la matriu en forma de gràfic de barres horitzontal, agafant com a variable comparativa la fila demanada a l'entrada de la funció. També ordena de major a menor.

Representació dels resultats - Correlacions de supervivència

```
correlations(["Pclass","Sex","Embarked","SibSp","Parch","L_Fare","L_Age","Title"], "Survived")
```





Amb un simple cop d'ull a les gràfiques, es pot respondre la pregunta inicial de les dades.

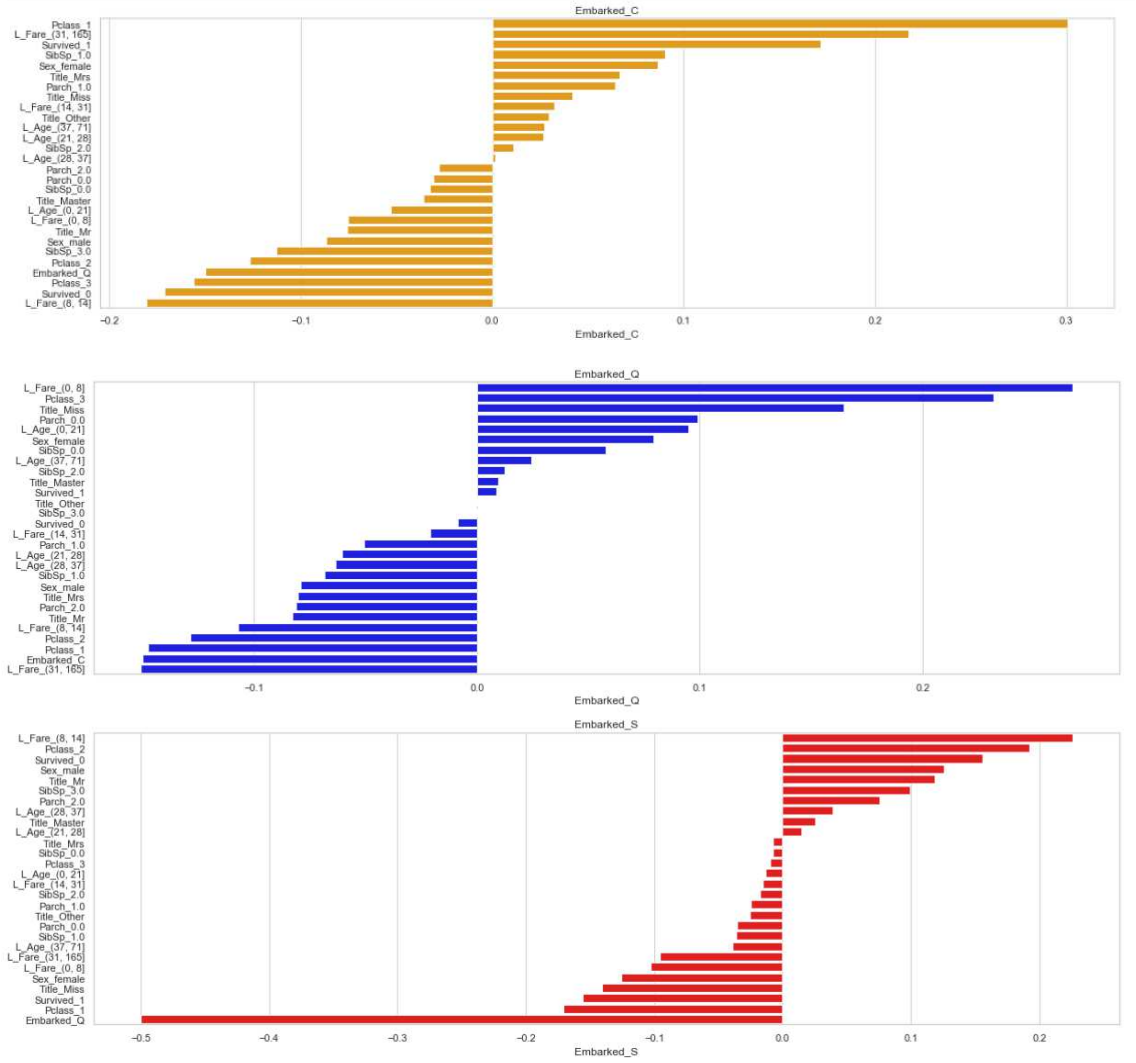
Qui va sobreviure? · Amb un cop d'ull a les gràfiques es veu que les següents característiques ajudaven a la supervivència:

1. Tenir classe 1 o 2.
2. Ser dona.
3. Haver embarcat al port 'C'
4. Tenir una tarifa cara (Fare).
5. Ser jove (menys de 21 anys).
6. Tenir 1 o 2 germans (SibSp).
7. Tenir 1 o 2 parents (Parch).

Aquesta no és la única pregunta que es pot respondre. De fet, és possible analitzar les correlacions de qualsevol variable.

· Altres preguntes que sorgeixen són: "Com es relaciona l'embarcament i la resta d'atributs? Per què embarcar per un lloc o altre ha de tenir influència en la supervivència?"

```
correlations(["Pclass","Sex","Survived","SibSp","Parch","L_Fare","L_Age","Title"],"Embarked")
```



D'aquí se'n treuen reflexions importants: l'embarcament va molt lligat a la tarifa, que al seu torn va lligat a la supervivència. A més, les dones tenien tarifes més altes i embarcaven més als ports 'C' i 'Q' i sabem que van sobreviure més dones. D'alguna manera, totes aquestes variables van lligades. És tracta de relacions de causalitat. Possiblement la gent de classe alta, de tarifa alta, i moltes de les dones embarcaven per 'C' o 'Q'. Potser a l'hora del rescat, són precisament aquestes persones les que van prioritzar en el salvament, d'aquí a que aquests embarcament mostrin alta probabilitat de supervivència.

Què més es pot analitzar?

Per acabar, s'aplicaran un parell de models per mirar de predir variables. Com s'ha vist, hi ha variables altament correlacionades entre elles, de manera que hauria de ser possible fer-ne prediccions precises. Cal adonar-se que una correlació, ja sigui molt positiva o molt negativa, és igualment útil per a fer prediccions. Les correlacions que no aporten cap mena d'informació són aquelles que són neutrals.

PREDICCIÓ DE LA CLASSE

L'algorisme utilitzar és un arbre de decisió simple. S'utilitzaran diverses opcions de fondària màxima de l'arbre, per veure quina dóna millors resultats.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

#Es preparen les dades
trax = X[["Sex","SibSp","Parch","Embarked","L_Age","L_Fare","Title"]]
tray = X[["Pclass"]]

tesx = X_test[["Sex","SibSp","Parch","Embarked","L_Age","L_Fare","Title"]]
tesy = X_test[["Pclass"]]

#Per fondària de l'arbre entre 2 i 15, els resultats són:
for x in range(2,15):
    regr = DecisionTreeClassifier(random_state=0, max_depth=x)
    regr.fit(trax, tray)
    result = regr.predict(tesx)
    acc = accuracy_score(tesy,result)
    print("El resultat per depth=",x,"és: ",acc*100,"%.")

print("\nEl millor resultat és del 78% per un arbre de fondària 9.")
print("Com s'ha vist els atributs estan altament relacionats i és viable fer-ne prediccions.")
```

```
El resultat per depth= 2 és:  63.63636363636363 %.
El resultat per depth= 3 és:  68.42105263157895 %.
El resultat per depth= 4 és:  72.00956937799043 %.
El resultat per depth= 5 és:  72.00956937799043 %.
El resultat per depth= 6 és:  72.96650717703349 %.
El resultat per depth= 7 és:  75.35885167464114 %.
El resultat per depth= 8 és:  74.64114832535886 %.
El resultat per depth= 9 és:  77.99043062200957 %.
El resultat per depth= 10 és:  77.51196172248804 %.
El resultat per depth= 11 és:  77.75119617224881 %.
El resultat per depth= 12 és:  77.51196172248804 %.
El resultat per depth= 13 és:  77.99043062200957 %.
El resultat per depth= 14 és:  77.99043062200957 %.

El millor resultat és del 78% per un arbre de fondària 9.
Com s'ha vist els atributs estan altament relacionats i és viable fer-ne prediccions.
```

Es possible predir la classe (1,2,3), a partir dels altres atributs, **sense tenir en compte la supervivència**, amb un 78% de precisió.

PREDICCIÓ DE L'EDAT

Una predicció encara més interessant seria predir l'edat ('Age'), una variable continua. Per això, és necessari un model de regressió. S'ha seleccionat un arbre de regressió.

Ara bé, és gairebé impossible endevinar de forma exacte una variable continua, caldrà dissenyar una funció de precisió pròpia per mesurar els resultats.

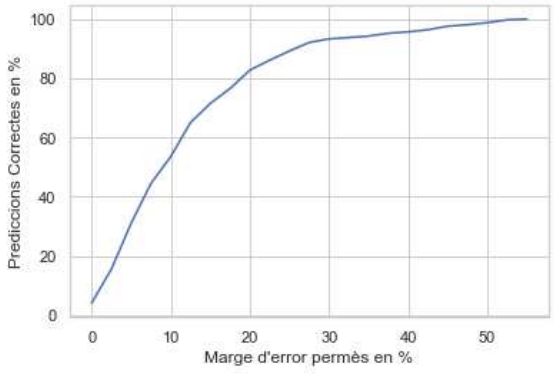
Òbviament, s'eliminarà la classe 'L_Age' que s'havia obtingut en l'enginyeria d'atributs. A més, igual que abans, tampoc es tindrà en compte la supervivència per a fer aquesta predicció.

```
dtr = DecisionTreeRegressor().fit(X.drop("L_Age",axis=1),train_age)
p_age = dtr.predict(X_test.drop("L_Age",axis=1))
```

```
#Per calcular la precisió no n'hi ha prou amb comparar els valors. Es provarà amb marges d'error de +-5% incrementanls:
l=[]
r=[]
print("Total d'exemples: ",len(test_age))
for x in range(0,23):
    count=0
    for real,predicted in zip(test_age,p_age):
        if ((predicted+(0.025*x*max(test_age))>= real)) and ((predicted-(0.025*x*max(test_age))<= real)):
            count+=1
    r.append(x)
    l.append(round(count*100/len(test_age),2))

predict_df = pd.DataFrame(l, columns=["Prediccions Correctes en %"])
predict_df["Marge d'error permès en %"] = [(x*2.5) for x in r]
sns.lineplot(x="Marge d'error permès en %", y="Prediccions Correctes en %", data=predict_df)
print(predict_df)
```

Per mesurar els resultats, s'han generat uns marges d'errors incrementals. Cada marge correspon a +-2.5% del valor màxim d'edat. D'aquesta manera, hi haurà un màxim de 40 marges d'error. La següent imatge mostra aquesta idea:

```
Total d'exemples:  418
    Prediccions Correctes en %  Marge d'error permès en %
0                         4.07                        0.0
1                        15.55                        2.5
2                        31.10                        5.0
3                        44.50                        7.5
4                        53.59                       10.0
5                        65.07                       12.5
6                        71.53                       15.0
7                        76.56                       17.5
8                        82.78                       20.0
9                        86.12                       22.5
10                       89.23                       25.0
11                       92.11                       27.5
12                       93.30                       30.0
13                       93.78                       32.5
14                       94.26                       35.0
15                       95.22                       37.5
16                       95.69                       40.0
17                       96.41                       42.5
18                       97.61                       45.0
19                       98.09                       47.5
20                       98.80                       50.0
21                       99.76                       52.5
22                      100.00                       55.0
```



Interpretació de la predicció d'edat

·Soprenentment, es pot predir l'edat amb un marge d'error del 20% en més del 80% dels casos.

·Per exemple, si l'edat d'algú era de 30 anys, en un 80% dels casos l'algorisme predirà una edat entre els 24 i els 36 anys (+-20%).

·El 100% de precisió s'obté amb un 55% marge d'error.

·Mirant la corba, el punt òptim entre millora i precisió seria en el 27.5% de marge d'error, on es predeix ja correctament el 92% dels casos.

6. Resolució del problema (conclusions)

El problema inicial era, donades unes dades de l'enfonsament del Titànic, mirar de descobrir **qui havia sobreviscut i qui no, i el per què**.

No només s'ha respost a aquesta pregunta, sinó que se n'han plantejat i respost d'altres d'igual d'interessants. S'ha intentat, per exemple, mirar d'entendre la relació entre la classe alta, l'embarcament, el preu de la tarifa i el sexe.

Finalment, s'han pogut predir altres variables com són l'edat o la classe, aprofitant l'alta correlació que s'havia detectat entre les variables.

A més, i com a extra, s'ha participat a la competició de *Kaggle* aplicant un algorisme classificador per la variable 'Survived'. El resultat ha estat satisfactori, dins el top 20%, tot i utilitzar un algorisme simple i sense ajust de hiperparàmetres. És una manera de confirmar que les decisions de neteja han estat efectives.