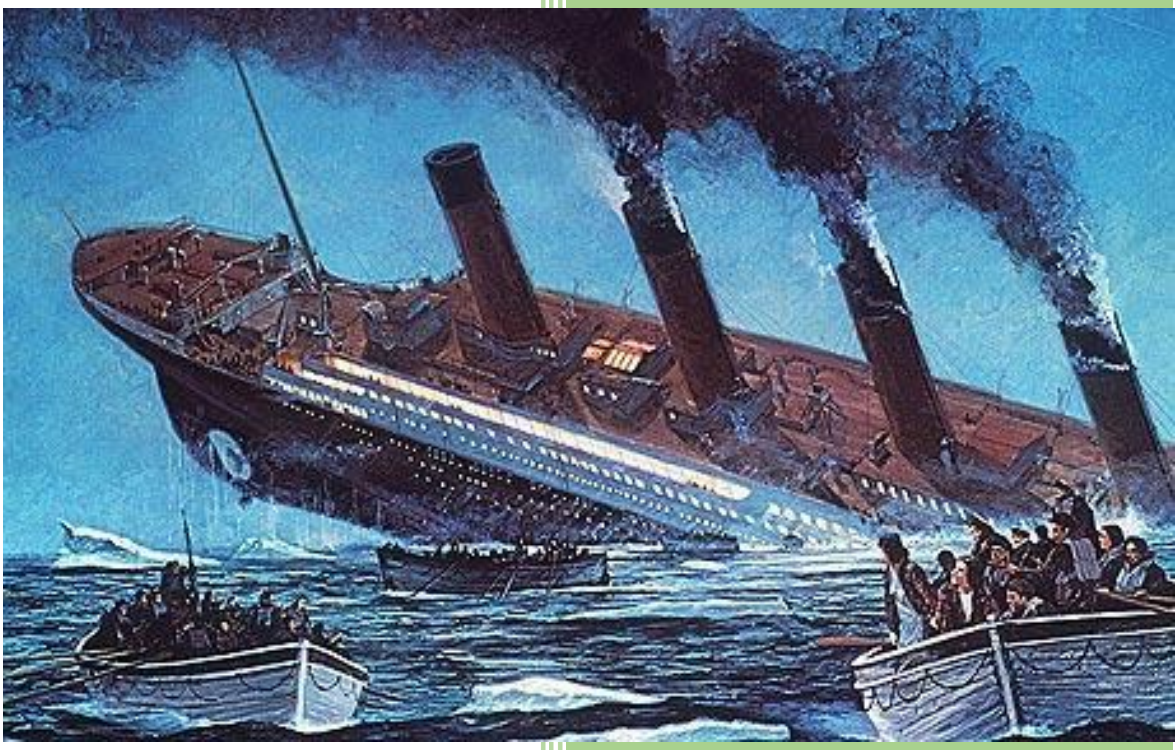
2020

Tipologia - Pràctica



Marcos Lacasa

17-5-2020

Índex

TITANIC

L'accident més important en un vaixell de la historia.

Va ser cap el 1912 quan en el viatge inaugural el transatlàntic més gran construït fins aquell moment xoca amb un iceberg a principis d'abril amb les aigües molt fredes. 2227 persones viatjaven en aquell moment però no es sap amb certesa el número de persones que hi havia en el moment de l'accident ja que va haver cancel·lacions d'última hora i també passatgers clandestins.

El xoc va ser a les 23.40h, en menys de 3 hores el Titanic era al fons del mar. A les 00.25 es va ordenar evacuar nens i senyores primer. La capacitat dels bots salvavides eren de 1100 persones, menys de la meitat dels que es suposa que hi havia en aquell moment. A més, les capacitats no s'omplien.

Botes	Hora del lanzamiento al mar	Capacidad	Contenido real	Lado	Personal de asistencia
Bote 1	01:05	40	12	Estribor	Murdoch, Lowe
Bote 2	01:45	40	18	Babor	Wilde, Smith
Bote 3	00:55	65	32	Estribor	Murdoch, Lowe
Bote 4	01:50	65	30	Babor	Lightoller
Bote 5	00:43	65	35	Estribor	Murdoch, Lowe, Pitman
Bote 6	01:10	65	24	Babor	Lightoller, Smith
Bote 7	00:40	65	28	Estribor	Murdoch, Lowe
Bote 8	01:00	65	28	Babor	Lightoller, Wilde, Smith
Bote 9	01:30	65	42	Estribor	Murdoch, Moody
Bote 10	01:50	65	34	Babor	Murdoch
Bote 11	01:35	65	55	Estribor	Murdoch
Bote 12	01:30	65	40	Babor	Wilde, Lightoller
Bote 13	01:40	65	64	Estribor	Murdoch, Moody
Bote 14	01:25	65	40	Babor	Lowe, Wilde, Lightoller
Bote 15	01:40	65	65	Estribor	Murdoch, Moody
Bote 16	01:20	65	40	Babor	Moody
Bote A	02:15	47	20	Estribor	Murdoch, Moody
Bote B	02:15	47	30	Babor	Lightoller
Bote C	02:00	47	40	Estribor	Murdoch, Wilde
Bote D	02:05	47	24	Babor	Lightoller, Wilde
Total		**1178**	**711**		

Il·lustració 1Font: https://es.wikipedia.org/wiki/Hundimiento_del_RMS_Titanic

Ens podem imaginar els moments crítics a l'hora d'omplir els bots. Es podien sentir els crits de socors mentre que centenars de persones morien en aigües a -1ºC. Es te constància dels debats que hi havia en els bots que no eren plens i que veien com altres morien al aigua, alguns van tornar a rescatar. Quaranta minuts més tard, un vaixell proper va arribar per rescatar a més persones.

Les xifres que tenim son aquestes:

	Primera clase	Segunda clase	Tercera clase	**Total de pasajeros**	Tripulación	**Total**
Hombres víctimas	118	154	387	**659**	670	**1329**
Mujeres víctimas	4	13	89	**106**	3	**109**
Niños víctimas	1	0	52	**53**	-	**53**
Total víctimas	**123** (37,8 %)	**167** (58,6 %)	**528** (74,8 %)	**818** (62,2 %)	**673** (76 %)	**1491** (67,7 %)
Hombres a bordo	175	168	462	**805**	862	**1667**
Mujeres a bordo	144	93	165	**402**	23	**425**
Niños a bordo	6	24	79	**109**	-	**109**
Total de personas a bordo	**325**	**285**	**706**	**1316**	**885**	**2201**

Il·lustració 2https://es.wikipedia.org/wiki/Hundimiento_del_RMS_Titanic

Van morir 818 passatgers dels 1316 que hi havia registres, i 673 dels 885 personal de tripulació. En total quasi el 70% de les persones que viatjaven al Titanic va morir.

La base de dades

Es pot descarregar a:

https://www.kaggle.com/c/titanic/data

I la informació que ens dona es la següent:

Un fitxer “train” i altre “test”.

Codi en python

El primer que hem de conèixer es què representa aquesta base de dades, anem a mirar si realment representa els viatgers del vaixell.

```
[3]: train.shape

[3]: (891, 12)

[4]: test.shape

[4]: (418, 11)
```

Tenim en total informació de 1309 passatgers, dels 1316 que es tenen constància de registres segons el quadre anterior.

Plantejament del problema

Ja podem plantejar el problema. Tenim un llistat de 1309 persones que viatjaven al vaixell en el moment de l'accident. Les dades s'han separat en un traint amb el 68% de les dades i amb un 32% restant en el test.

El repte

Podem treballar un model que pugui predir qui va sobreviure i qui no?

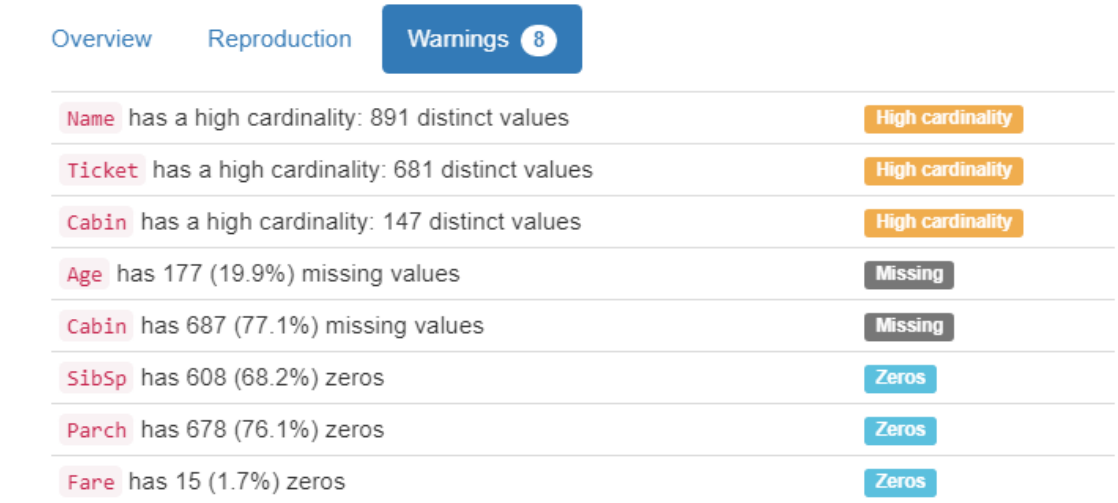
Visualització ràpida

Per una visualització ràpida i un informe acurat podem treballar amb la llibreria

```
from pandas_profiling import ProfileReport
```

Analitzem només l'arxiu traint per preparar el nostre model.

Els primers avisos que ens ofereix l'informe es:



Integració i selecció de les dades d'interès a analitzar.

Quines variables ens ofereix la base de dades de cada passatger?

```
[5]: train.head()
```

[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

PassengerId – Es un nº identificació de la base de dades per cada passatger.

Survived – 0 (no va sobreviure), 1 (SI va sobreviure).

Pclass – A quina classe va viatjar. Hi ha 1,2,3 classes. La 1a classe la més luxosa.

Name – Indica el nom, i un títol, com Mr, Miss ... els noms entre parèntesi no es repeteix en la base de dades train. Aquí ens indica el sexe i l'estat, si és casada o no.

SibSP.- El número de germans, cosins que viatgen. Número.

Parch – Si el pare o mare viatjava. Podrien estar els 2.

Ticket – Número de ticket, hi havia compres conjuntes, més d'un passatger viatjava en el mateix ticket.

Fare - El cost de la compra. Quan es va pagar per ticket.

Cabin – Le nom de la cabina.

Embarked – La ciutat on va embarcar el passatger.

3 Neteja de les dades

3.1 Elements buits

3.1.1 Age

```
[4]: #missing values
     train.isna().sum()
```

```
[4]: PassengerId      0
     Survived         0
     Pclass           0
     Name             0
     Sex              0
     Age            177
     SibSp            0
     Parch            0
     Ticket           0
     Fare             0
     Cabin          687
     Embarked         2
     dtype: int64
```

Només tenim 3 variables amb missing, 2 d'elles amb números importants, com son l'edat i la Cabina. Com ho podem tractar?

La Cabina es francament difícil. Però com sabem a quina classe viatjava, sabem si era una cabina luxosa o no. També sabem si viatjava sola o no. La variable Cabin es una variable que no ens preocupa gaire.

L'Edat ja és més complexa. De totes les tècniques que hi ha per imputar valors (el que no farem es eliminar 177 passatgers) hem de treballar una que sigui el més eficaç possible. Com la majoria de les variables son categòriques, algorismes per regressió no series eficaços.

Com podem relacionar l'edat amb les variables que tenim.

[6]:
```
#Ara tenim una base de dades amb 177 missing d'edat de 890 passatgers
#Podriem intuir edat per classe?
train['Age'].groupby([train['Pclass']]).mean()
```

[6]:
```
Pclass
1    38.233441
2    29.877630
3    25.140620
Name: Age, dtype: float64
```

Hi ha una relació entre Edat i classe evident. Ja tenim una pista. Podem aprofundir més.

El que he fet és crear una nova variable que es diu 'state' i que consisteix en 4 etiquetes que dependrà del títol que indica en el Nom:

Master – si indica Master.

Miss – si indica Miss, Mlle o Ms.

Mr – si indica Mr, Don, Major, Capt, Jonkheer, Rev, Col o Dr.

Mrs – Si indica Mrs, Countess, Mme

Ara anem a calcular dades estadístiques però per 'Pclass' i 'state'.

[11]:
```
#Arribat a aquest punt podem imputar els valors missing en l'edat, però ho farem amb les mitjanes dels grups que pertànyin.
#Primer calculem la edat mitja dels grups que creem
train.groupby(['Pclass', 'state'])[['Age']].describe()
```

[11]:

		Age							
		count	mean	std	min	25%	50%	75%	max
Pclass	state								
1	Master	3.0	5.306667	5.165475	0.92	2.46	4.0	7.500	11.0
	Miss	46.0	29.869565	12.739455	2.00	21.25	30.0	36.000	63.0
	Mr	99.0	42.449495	13.890699	17.00	31.00	42.0	51.000	80.0
	Mrs	38.0	39.973684	12.656559	17.00	33.50	39.5	49.750	62.0
2	Master	9.0	2.258889	2.342634	0.67	0.83	1.0	3.000	8.0
	Miss	30.0	23.116667	13.569542	2.00	14.00	24.5	31.875	50.0
	Mr	91.0	33.307692	12.481491	8.00	24.50	31.0	39.000	70.0
	Mrs	43.0	33.116279	10.413383	14.00	24.50	31.0	40.500	57.0
3	Master	24.0	5.350833	3.593608	0.42	2.00	4.0	9.000	12.0
	Miss	69.0	16.123188	9.697315	0.75	9.00	18.0	22.000	45.0
	Mr	229.0	28.724891	10.490946	11.00	21.00	26.0	34.000	74.0
	Mrs	33.0	33.515152	10.031579	15.00	27.00	31.0	40.000	63.0

Ara si veiem una diferència significativa entre l'edat per class i per State.

Estudiem la estructura dels registres on NO tenim la variable 'Age'.

```
#Els 177 missing, distribució
pd.crosstab([ Edad.Pclass, Edad.state], Edad.Survived, margins=True)
```

	Survived	0	1	All
Pclass	**state**			
1	**Miss**	0	1	1
	Mr	16	5	21
	Mrs	0	8	8
2	**Miss**	0	2	2
	Mr	7	2	9
3	**Master**	2	2	4
	Miss	14	19	33
	Mr	83	7	90
	Mrs	3	6	9
All		125	52	177

Així doncs podem imputar a cada grup l'edat mitjana. Per exemple:

Classe 1 – Miss imputem 29.8 -> 30 anys

La majoria dels registres son a la classe 3, per la qual cosa hi ha un biaix important i per això pot variar el resultat del model en funció de la imputació de l'edat.

3.1.2 Cabin

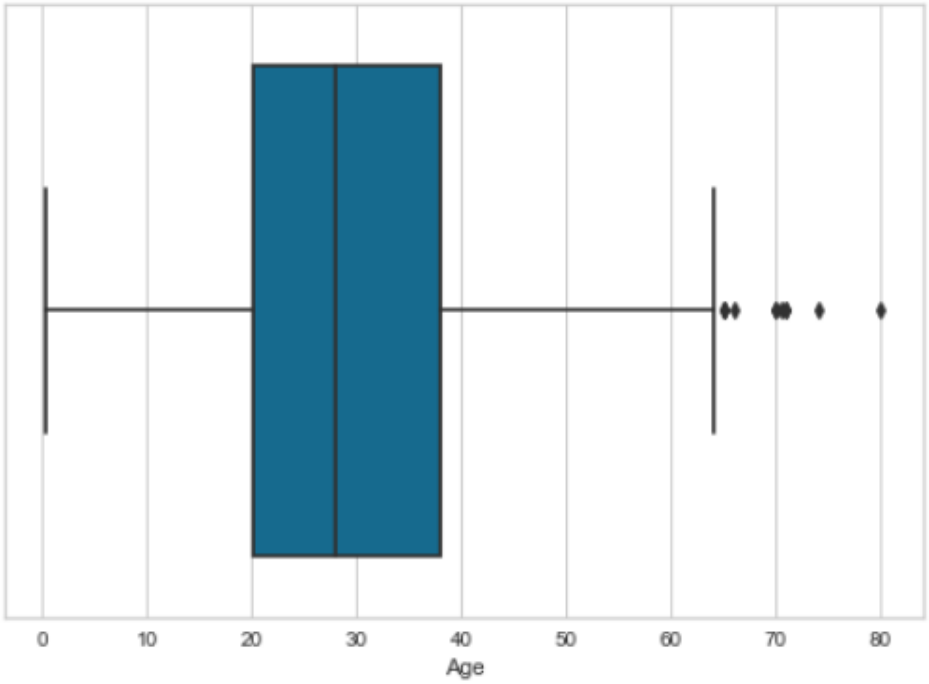
No treballarem aquesta variable donat que hi ha massa percentatge de valors buits. La informació d'on viatjaven es pot deduir del numero consecutiu de ticket, Fare i Pclass.

Identificació de valors extrems

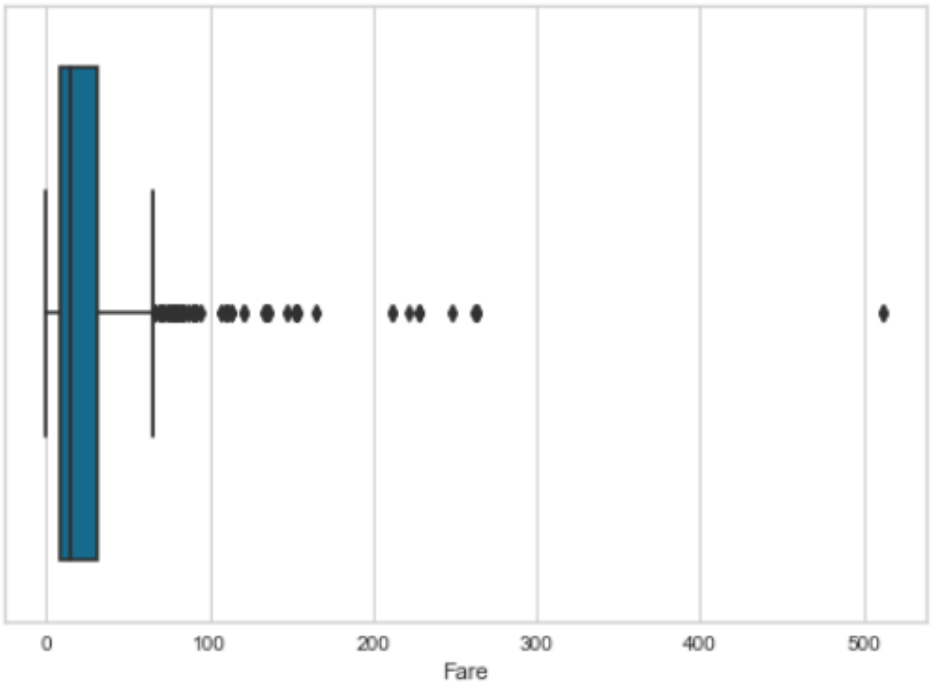
De les variables a estudiar, només dos tindrien valors numèrics ('Age' i 'Fare').

Boxplot

```
[32]: ax = sns.boxplot(x="Age", data=train)
```




```
[33]: ax = sns.boxplot(x="Fare", data=train)
```



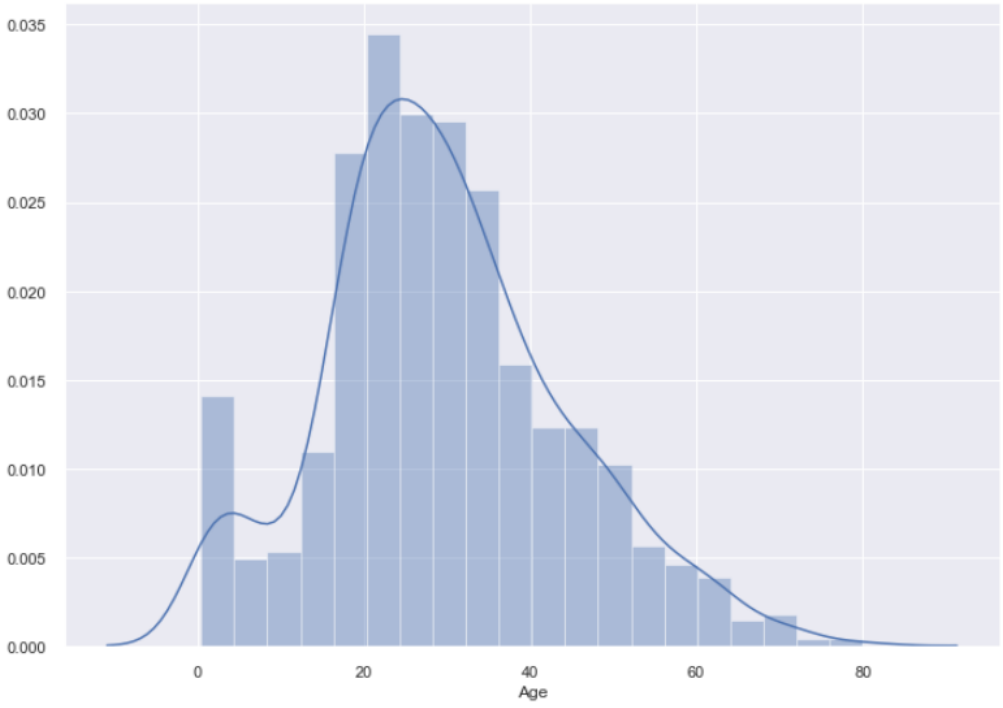
Aquí veiem clarament que l'edat es concentra entre els 20-40 anys. Per això ha estat important la imputació per grups sota la meva opinió. En canvi el cost tal i com havia detectat, representa un cost de pagament per diferents tickets. No seria bó imputarlo al mateix passatger ja que correspon a més d'un.

Histogrames:

Age

```
[9]: #Discretitzar edad, histograma
     sns.set(color_codes=True)
     fig, ax = plt.subplots()
     fig.set_size_inches(11.7, 8.27)
     sns.distplot(train['Age'])
```

```
[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8d5ca5048>
```

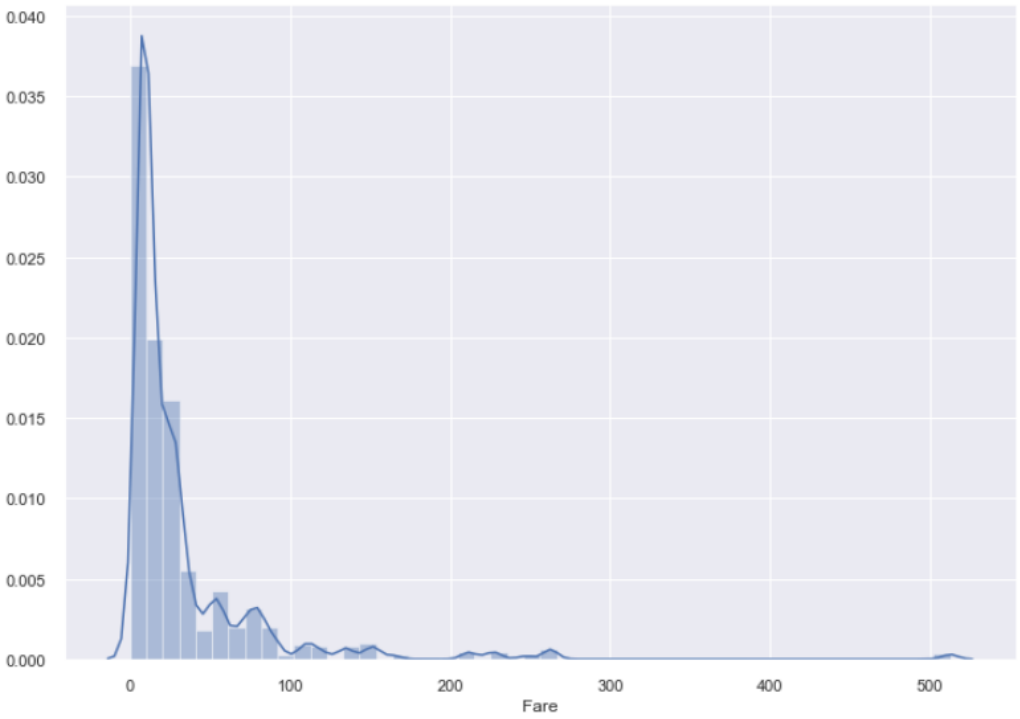


Amb una clara kurtosis a l'esquerra indica la concentració entre la població entre 20-40 amb un pic entre nadons i menors de 5 anys.

Fare

```
[10]: sns.set(color_codes=True)
      fig, ax = plt.subplots()
      fig.set_size_inches(11.7, 8.27)
      sns.distplot(train['Fare'])
```

```
[10]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8d5cfe7c8>
```



La gran concentració entre els valors econòmics, que tindrà un efecte directe amb el tipus de classe que viatjaven.

```
[11]: train['Pclass'].groupby([train['Pclass']]).count()
```

```
[11]: Pclass
      1    216
      2    184
      3    491
      Name: Pclass, dtype: int64
```

```
[15]: train['Fare'].groupby([train['Pclass']]).mean()
```

```
[15]: Pclass
      1    84.154687
      2    20.662183
      3    13.675550
      Name: Fare, dtype: float64
```

```
[22]: train['Fare'].groupby([train['Pclass']]).sum()
```

```
[22]: Pclass
      1    18177.4125
      2     3801.8417
      3     6714.6951
      Name: Fare, dtype: float64
```

Aquestes dades, les podem presentar millor amb un quadre:

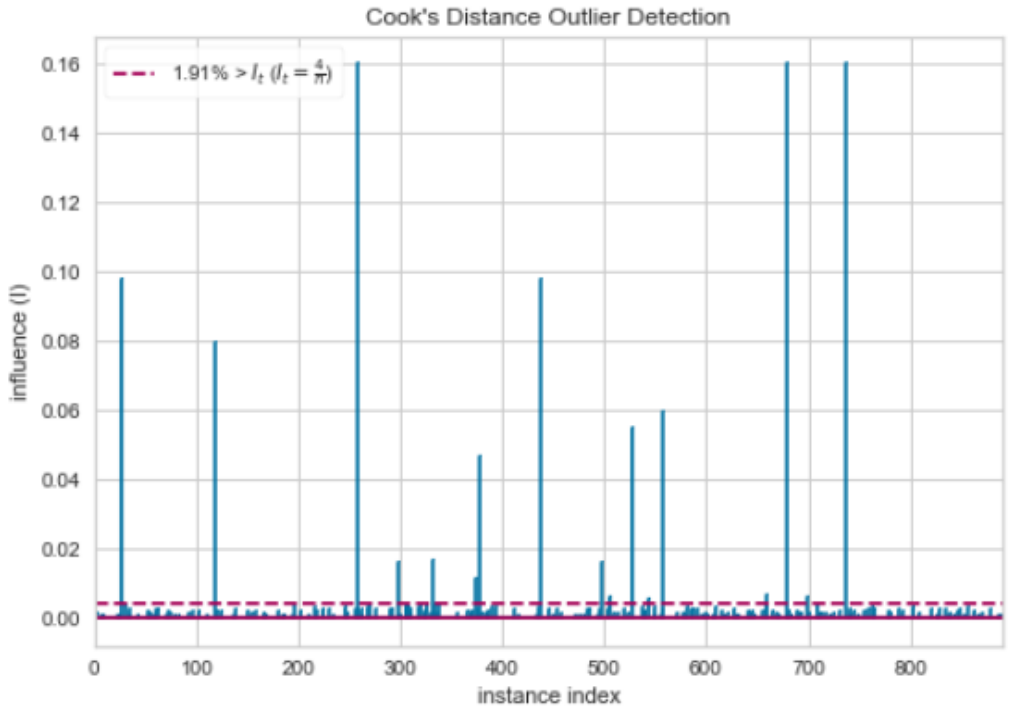
[45]:

Pclass	numero	despesaTotal	mitja	cum_sumNum	cum_sumTot	percNum	percTot
1	216	18177.4125	84.154687	216	18177.4125	24.242424	63.349288
2	184	3801.8417	20.662183	400	21979.2542	44.893378	76.598916
3	491	6714.6951	13.675550	891	28693.9493	100.000000	100.000000

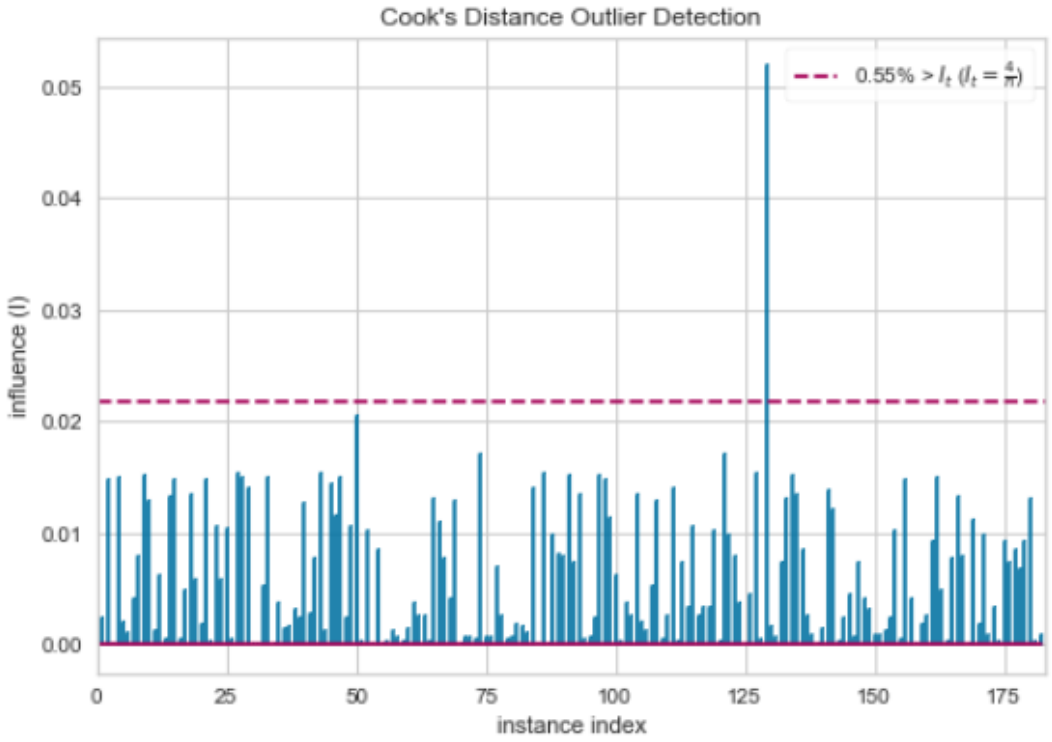
On veiem que el 24% dels passatgers representen una despesa del 63%.

La distància de cook es una mesura per identificar els valors fora de rang. Per això utilitzem la llibreria tellowbrick.

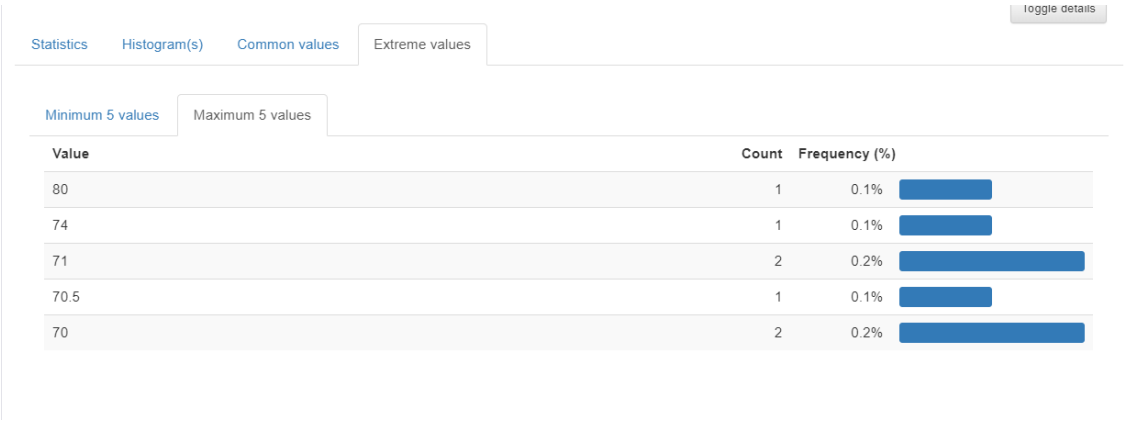
```
from yellowbrick.regressor import CooksDistance
from yellowbrick.datasets import load_concrete

X=np.asarray(train['Fare']).reshape(-1,1)
y=np.asarray(train['Survived'])
visualizer = CooksDistance()
visualizer.fit(X,y)
visualizer.show()
```



Tot son compres de classe 1, molt per sobre de la mitja. Aquestes diferències tan grans les expliquem a que hi havia unes cabines luxoses. Això ho hem de tenir en compte per avaluar si volem un model amb un Fare amb valors continus, o un Fare discretitzat. Tot i que hi haurà una relació molt estreta Fare-Pclass, podríem discriminar. Ho estudiem després.



Només un sol individu, mirem al gràfic que em imprès abans els valors màxims:



No farem res, entenem que una persona de 80 anys no alterarà l'avaluació del model, a més valorem discretitzar l'edat.

4. Anàlisi de les dades

4.1 Selecció dels grups de dades que es volen analitzar/comparar (planificació dels anàlisis a aplicar).

4.1.1 Variable Fare

Si busquem una relació entre la supervivència i el sexe + Pclass tenim una evidència molt interessant:

[50]:
```
#Afegim classe
pd.crosstab([train.Sex, train.Survived], train.Pclass, margins=True, normalize='columns')
```

[50]:

Sex	Survived	Pclass 1	2	3	All
female	0	0.013889	0.032609	0.146640	0.090909
	1	0.421296	0.380435	0.146640	0.261504
male	0	0.356481	0.494565	0.610998	0.525253
	1	0.208333	0.092391	0.095723	0.122334

Pràcticament totes les dones de classe 1 i 2 van sobreviure. I el percentatge més important dels homes es concentra a la classe 1. La relació entre supervivència, ser dona i la classe en que viatjava és molt alta.

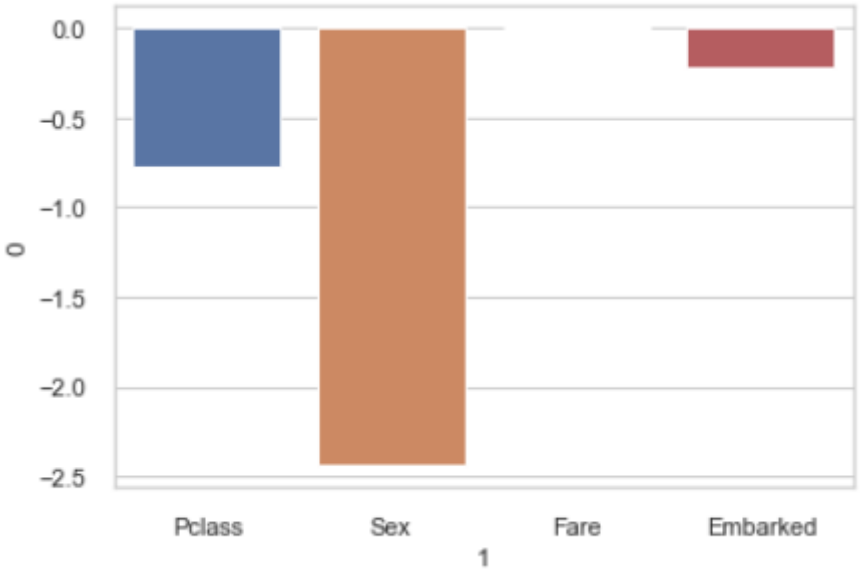
Ho podem mesurar:

Creo una regressió logística amb els paràmetres: Pclass, Sex, Fare i Embarked.

```
print(classification_report(y_log, reglog_model.predict(X_log)))
```

```
              precision    recall  f1-score   support

           0       0.81      0.84      0.83       549
           1       0.73      0.69      0.71       342

    accuracy                           0.78       891
   macro avg       0.77      0.76      0.77       891
weighted avg       0.78      0.78      0.78       891
```

Es una bona precisió, tenint en compte que estem lluny d'afinar el model. Només volem una primera aproximació sobre la importància d'algunes variables.



Provem discretitzar 'Fare' per veure la diferència, tot i que la relació directa entre Pclass és evident. Com que Pclass hi ha només 3 classes, discretitzarem Fare en 5 trams.

[189]:
```
#Discretitzem Fare amb cut, treballarem 10 trams que cada tram no te perquè tenir els mateixos integrants.
reglog['Fare']=train['Fare']
reglog['Fare_d']=pd.qcut(reglog['Fare'], 5, labels=range(5))
pd.crosstab(reglog.Fare_d, reglog.Survived, margins=True, normalize='columns')
```

[189]:

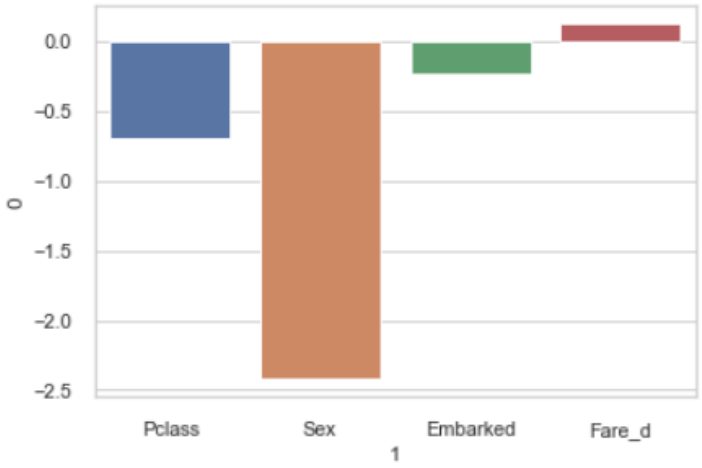
Survived	0	1	All
Fare_d			
0	0.255009	0.114035	0.200898
1	0.267760	0.108187	0.206510
2	0.180328	0.213450	0.193042
3	0.182149	0.233918	0.202020
4	0.114754	0.330409	0.197531

[190]:
```
#Tornem a avaluar el model, ara substituim la variable Fare_d x Fare
reglog['Fare']=train['Fare']
reglog['Fare_d']=pd.qcut(reglog['Fare'], 5, labels=range(5))
reglog=reglog.drop(['Fare'], axis=1)
X_log = reglog.drop(['Survived'], axis=1)
y_log = reglog['Survived']
reglog_model = LogisticRegression(solver='liblinear', random_state=0)
reglog_model.fit(X_log, y_log)
print(classification_report(y_log, reglog_model.predict(X_log)))
```

```
              precision    recall  f1-score   support

           0       0.82      0.83      0.82       549
           1       0.72      0.71      0.72       342

    accuracy                           0.78       891
   macro avg       0.77      0.77      0.77       891
weighted avg       0.78      0.78      0.78       891
```

Avaluem ara el model a veure si incrementa la precisió i la importància de la variable.

[191]:
```
importance = reglog_model.coef_[0]
var_imp = pd.DataFrame(zip(importance,X_log.columns.values ))
sns.set(style='whitegrid')
ax = sns.barplot(x= var_imp[1], y= var_imp[0], data=var_imp)
```



Conclusions: No ha variat gaire, la precisió es molt similar, 0.78. Ha augmentat la importància de Fare_d, però Sex es realment crític. Avaluem l'edat.

4.1.2 Variable Age

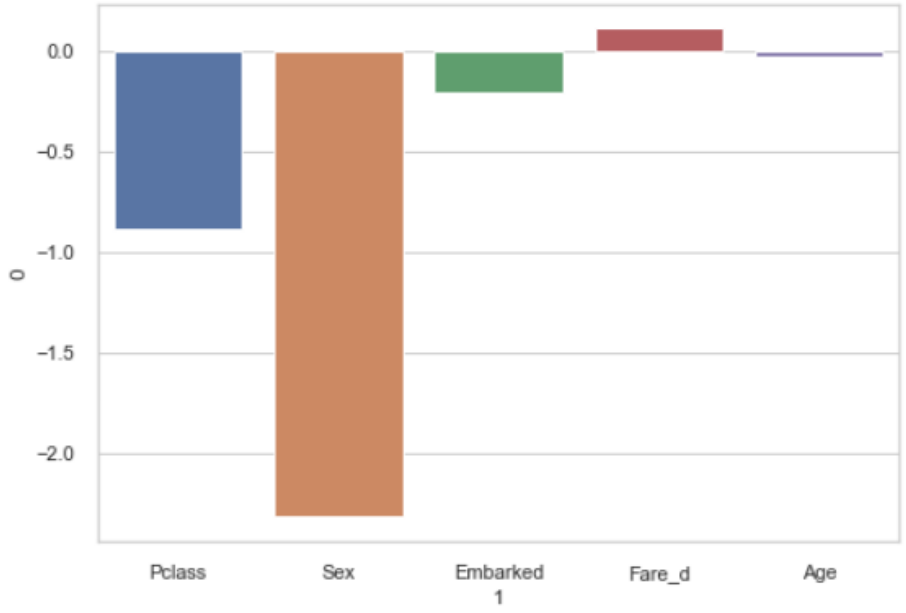
Avaluem la regressió per conèixer la importància de les variables.

Utilitzem la variable contínua Age.

[68]:
```
#Avaluem age amb el model de regressió
reglog['Age']=train['Age']
reglog['edad']=pd.qcut(reglog['Age'], 7, labels=range(7))
reglog=reglog.drop(['Age'], axis=1)
X_log = reglog.drop(['Survived'], axis=1)
y_log = reglog['Survived']
reglog_model = LogisticRegression(solver='liblinear', random_state=0)
reglog_model.fit(X_log, y_log)
print(classification_report(y_log, reglog_model.predict(X_log)))
```

```
              precision    recall  f1-score   support

           0       0.82      0.85      0.84       549
           1       0.75      0.70      0.72       342

    accuracy                           0.79       891
   macro avg       0.78      0.78      0.78       891
weighted avg       0.79      0.79      0.79       891
```

[69]:
```
importance = reglog_model.coef_[0]
var_imp = pd.DataFrame(zip(importance,X_log.columns.values ))
sns.set(style='whitegrid')
ax = sns.barplot(x= var_imp[1], y= var_imp[0], data=var_imp)
```



Incrementem la precisió en 1 punt, passem de 0.78 a 0.79, embarked i la variable Pclass ha augmentat la seva importància reduint una mica Sex.

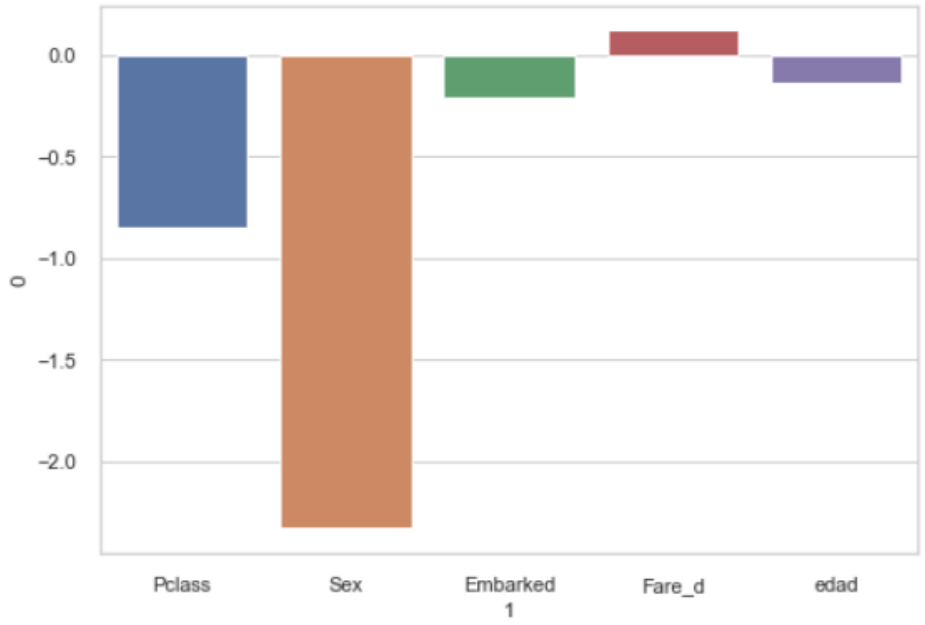
Estudiem la diferència si discretitzem Age, creem una variable que sigui ‘edad’.

Un cop analitzat, 7 trams és un valor que millora la precisió en altre punt, ja arribem al 0.8

[70]:
```
#Avaluem age amb el model de regressió
reglog['Age']=train['Age']
reglog['edad']=pd.qcut(reglog['Age'], 7, labels=range(7))
reglog=reglog.drop(['Age'], axis=1)
X_log = reglog.drop(['Survived'], axis=1)
y_log = reglog['Survived']
reglog_model = LogisticRegression(solver='liblinear', random_state=0)
reglog_model.fit(X_log, y_log)
print(classification_report(y_log, reglog_model.predict(X_log)))
```

```
              precision    recall  f1-score   support

           0       0.82      0.86      0.84       549
           1       0.75      0.70      0.73       342

    accuracy                           0.80       891
   macro avg       0.79      0.78      0.78       891
weighted avg       0.80      0.80      0.80       891
```

[71]:
```
importance = reglog_model.coef_[0]
var_imp = pd.DataFrame(zip(importance,X_log.columns.values ))
sns.set(style='whitegrid')
ax = sns.barplot(x= var_imp[1], y= var_imp[0], data=var_imp)
```



A part del punt de precisió, veiem com ‘edad’ ha guanyat en importància i això és important per garantir la bondat del model.

Conclusions.- La discretització de ‘Age’ millora el model.

4.2 Normalitat i Homogeneïtat de la variància.

4.1.1 Age

Tenim dos mostres a avaluar:
Age- Es una variable on coneixem les mitjes i variàncies poblacionals. Llavors podem avaluar la distribució de la mostra, i encara que no sigui normal, com la mostra és suficientment gran (n> 50), podríem utilitzar estadístics paramètrics.

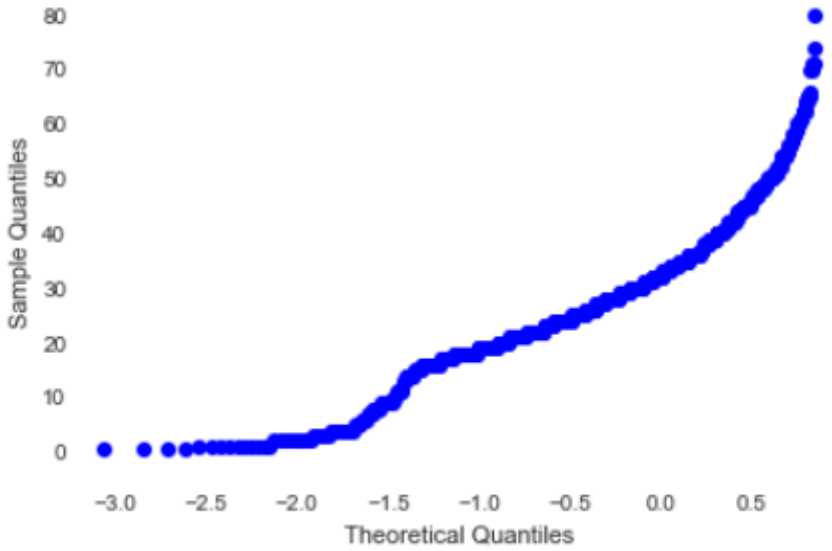
Per avaluar la normalitat de la mostra Age utilitzem l'estadístic de Kolmogorov-Smirnov.

[7]:
```
#Estudi de normalitat en Age i Fare
#Per mostres més grans de 50 Kolmogorov-Smirnov, mitja coneguda poblacional.

stats.kstest(train['Age'], 'norm')
```

[7]:
```
KstestResult(statistic=nan, pvalue=nan)
```

Es a dir, una mostra amb una distribució no normal.
Per il·lustrar-ho amb un qqplot

[20]:
```
# Anàlisi visual, distribució dels quantils
from statsmodels.graphics.gofplots import qqplot

qqplot(train['Age'], line='s')
plt.show()
```



Aquestes dades justifica la discretització de la variable.

4.1.2 Fare

Amb la variable 'Fare' utilitzarem l'estadístic de normalitat Liliefors ja que no coneixem els paràmetres poblacionals, a diferència de l'edat. No podem dir que la mostra prové d'una població amb distribució normal.

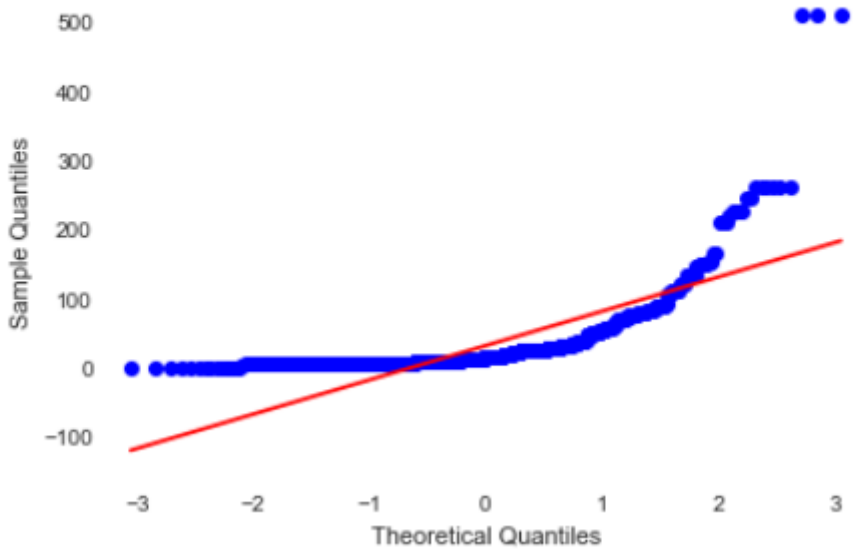
```
[11]: #Mitja poblacional no coneguda, Lilliefors
      from statsmodels.stats.diagnostic import lilliefors

      lilliefors(train['Fare'], dist='norm', pvalmethod=None)
```

```
[11]: (0.28184804098597455, 6.069408967120122e-202)
```

I tampoc ens ofereix una distribució normal. Amb una p-value molt inferior a 0.05.

Avaluem gràficament la distribució amb una gràfica de distribució de quantils.

```
[21]: qqplot(train['Fare'], line='s')
      plt.show()
```



Per la qual cosa, queda totalment justificada la discretització de la variable. La gràfica ens mostra una distribució molt allunyada d'una línia de 45º, com era de preveure donat el seu histograma.

4.3 Comparació de grups de dades

4.3.1 Comparació de classificadors

Comparar 9 classificadors avaluant els hiperparàmetres per una millor precisió.

Passes:

1- Seleccionar 9 classificadors.
2- Avaluar els hiperparàmetres per definir els models amb millors resultats mitjançant un GridSearch.
3- Crear un Pipeline amb els models definits.

4.3.1.1 GridSearch tunning dels models

Adjunto un exemple del codi para avaluar cadascun dels 9 models analitzats:

[87]:
```
#GridSearch KNeighborsClassifier, ni millora 0.77
from sklearn.metrics import accuracy_score, log_loss
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC, LinearSVC, NuSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis

kn = Pipeline(steps=[('preprocessor', preprocessor),
                     ('classifier', KNeighborsClassifier())])
param_grid = {
    'classifier__n_neighbors': [11,13,15],
    'classifier__leaf_size': [1,2,3],
    }
from sklearn.model_selection import GridSearchCV
GSkn = GridSearchCV(kn, param_grid, n_jobs= 1)

GSkn.fit(X_train, y_train)
print(GSkn.best_params_)
print(GSkn.best_score_)
print("model score: %.3f" % GSkn.score(X_test, y_test))
```

```
{'classifier__leaf_size': 2, 'classifier__n_neighbors': 13}
0.8323644933228594
model score: 0.771
```

En aquest cas s'avalua el classificador KNN (KneighborsClassifier) amb dos paràmetres i concretem la definició del model amb una precisió esperada.

El codi ens retorna els paràmetres que ofereixen millors resultat. En aquest cas:

El paràmetre 'leaf_size' avaluar 2.

El paràmetre 'n_neighbors' millor valor 13.

Dels subgrups de dades analitzats, ha arribat a un 0.83 de precisió màxima, sent la mitjana del 0.771.

4.3.2 Chi quadrat

Per avaluar la influència de paràmetres, ho hem fet des de la regressió lineal, però també ho podem fer mitjançant les taules de contingència amb el Chi Quadrat.

4.3.2.1 Pclass vs Supervived

Utilitzarem la llibreria searchpy.

Ens ofereix, tal i com explico al codi, bones possibilitats de treballar les taules de contingència.

Adjunto els resultats i la explicació:

```
[104]: #Però també podem avaluar l'anàlisi d'independència amb chi quadrat directament.
       researchpy.crosstab(train['Survived'], train['Pclass'], prop='col', test= "chi-square")
```

```
[104]: (          Pclass
        Pclass         1       2       3     All
        Survived
        0          37.04   52.72   75.76   61.62
        1          62.96   47.28   24.24   38.38
        All       100.00  100.00  100.00  100.00,
                        Chi-square test   results
        0  Pearson Chi-square ( 2.0) =   102.8890
        1                   p-value =     0.0000
        2                Cramer's V =     0.3398)
```

La interpretació del resultat

Ens ofereix el quadre de contingència amb les proporcions per columnes.

Pearson Chi-square (2.0) = 102.8890 es el valor de l'estadístic.

p-value = 0.0000 indica la significància de la independència de les variables. p<0.05 hi ha una dependència.

Cramer's V = 0.3398 per valors majors a 0.5 indica una associació forta. Amb valors 0.3-0.5 associació moderada.

4.3.2.2 Taula amb totes les variables

Adjunto la taula amb el codi.

```
[191]: #Preparem les taules
       categorical_features = train.select_dtypes(include=['object']).columns
       taula=[]
       pvalue=[]
       cramer=[]
       for i in categorical_features:
           res = researchpy.crosstab(train['Survived'], train[i], prop='col', test='chi-square')
           taula.append(res[0])
           pvalue.append(res[1].iloc[1])
           cramer.append(res[1].iloc[2])
```

```
[192]: cramer=pd.DataFrame(cramer)
       cramer['variables'] = categorical_features
       cramer
```

[192]:

	Chi-square test	results	variables
2	Cramer's V =	0.3398	Pclass
2	Cramer's phi =	0.5434	Sex
2	Cramer's V =	0.1726	Embarked
2	Cramer's V =	0.5680	state
2	Cramer's V =	0.2139	edad
2	Cramer's V =	0.3357	Fare_d

Les variables amb una dependència més forta amb la supervivència son:

Sex – Amb un índex de Cramer (única taula 2x2) de 0.54

State – amb un índex de 0.568.

5. Representació dels resultats

Un cop parametritzats els classificadors, tenim una taula amb els resultats:

[96]:
```
Resultats
```

[96]:

	Models	Scores
0	KNeighborsClassifier	0.784753
1	SVC	0.798206
2	NuSVC	0.825112
3	DecisionTreeClassifier	0.780269
4	RandomForestClassifier	0.811659
5	AdaBoostClassifier	0.802691
6	GradientBoostingClassifier	0.784753
7	LogisticRegression	0.834081
8	XGBClassifier	0.793722

Hem arribat a un 0.834 de precisió amb el model de LogisticRegression.

Podem millorar?

5.1 Stacking

Com tenim totes les prediccions de tots els classificadors, podem iniciar un model amb Stacking. Escollim, per crear la dataset les variables amb millor score.

El modelem amb LogisticRegression.

[209]:
```
y_pred = LRs.predict(meta)
print('Final prediction score: [%.8f]' % accuracy_score(y_test, y_pred))
```

```
Final prediction score: [0.83856502]
```

El resultat final millora, passa de 0.834 a 0.8356

Es una petita millora, però ens prepara pel següent i últim model.

5.2 Cascading

Afegim la dataset de Stacking al traint dataset, tècnica anomenada Cascading.

El modelem amb LogisticRegression.

Resultat:

[215]:
```
LRc.fit(meta1, y_test)
y_pred = LRc.predict(meta1)
print('Final prediction score: [%.8f]' % accuracy_score(y_test, y_pred))
```

```
Final prediction score: [0.85201794]
```

Millorem la precisió amb 2 punts. Passem del 0.8385 a un 0.852

6. Resolució del problema. A partir dels resultats obtinguts, quines són les conclusions? Els resultats permeten respondre al problema?

Aquesta taula pot resumir molt bé el resultat final

```
[230]: #Finalment amb cascading augmento score de 0.83 a 0.85
       print(classification_report(y_test, LRc.predict(meta1)))
```

	precision	recall	f1-score	support
0	0.85	0.90	0.88	130
1	0.85	0.78	0.82	93
accuracy			0.85	223
macro avg	0.85	0.84	0.85	223
weighted avg	0.85	0.85	0.85	223

Això vol dir que amb el millor model trobat, podem predir amb un 88% de probabilitats qui no sobreviurà a l'accident i amb un 82% de probabilitats, qui sí sobreviurà.

També sabem que el fet de ser dona i el fet de ser casada o no i comprar un bitllet de primera classe, influeix decisivament en el fet de sobreviure.

7. Codi python, notes sobre la originalitat del problema

HI ha molts models sobre Titanic a la xarxa, per això vull remarcar alguns passos que no he trobat enlloc.

1- Mètodes de discretització.
   a. Tant en el mètode Fare com Age he creat un mètode per discretitzar, i sobre tot binaritzar que és diferent al que hi ha publicat. No dic que ni millor ni pitjor, però sí diferent.
2- Avaluació variables més influents mitjançant Chi Quadrat, especialment amb el codi emprat.
3- Ús de tècniques com Stacking o Cascading per millorar la precisió.