Descripció

Familiaritza't amb la programació científica mitjançant la llibreria SKLearn / Scikitlearn.

In [26]:
```
import pandas as pd

from sklearn.model_selection import train_test_split
```

Importación del dataset

In [27]:
```
pd.set_option('max_columns', None)
raw_df = pd.read_csv('DelayedFlights.csv', sep=",", encoding='utf8')
```

In [28]:
```
raw_df.head()
```

Out[28]:

	Unnamed: 0	Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarrier	FlightNum	TailNum
0	0	2008	1	3	4	2003.0	1955	2211.0	2225	WN	335	N712SW
1	1	2008	1	3	4	754.0	735	1002.0	1000	WN	3231	N772SW
2	2	2008	1	3	4	628.0	620	804.0	750	WN	448	N428WN
3	3	2008	1	3	4	1829.0	1755	1959.0	1925	WN	3920	N464WN
4	4	2008	1	3	4	1940.0	1915	2121.0	2110	WN	378	N726SW

Nivell 2

- **Exercici 2:**

Aplica algun procés de transformació (estandarditzar les dades numèriques, crear columnes dummies, polinomis...).

Se comprueba la cantidad de datos NaN incluidos en el dataset

In [29]:
```
raw_df.isna().sum()
```

Out[29]:
```
Unnamed: 0                0
Year                      0
Month                     0
DayofMonth                0
DayOfWeek                 0
DepTime                   0
CRSDepTime                0
ArrTime                7110
CRSArrTime                0
UniqueCarrier             0
FlightNum                 0
TailNum                   5
ActualElapsedTime      8387
CRSElapsedTime          198
AirTime                8387
ArrDelay               8387
DepDelay                  0
Origin                    0
Dest                      0
Distance                  0
TaxiIn                 7110
TaxiOut                 455
Cancelled                 0
CancellationCode          0
Diverted                  0
CarrierDelay         689270
WeatherDelay         689270
NASDelay             689270
SecurityDelay        689270
LateAircraftDelay    689270
dtype: int64
```

Para no eliminar un número considerable de datos que nos interesan, se reemplaza el valor NaN por el valor medio de cada columna.

In [30]:
```
raw_df['ArrDelay']=raw_df['ArrDelay'].fillna(raw_df['ArrDelay'].mean())
```

In [31]:
```
raw_df['CarrierDelay']=raw_df['CarrierDelay'].fillna(raw_df['CarrierDelay'].mean())
```

In [32]:
```
raw_df['WeatherDelay']=raw_df['WeatherDelay'].fillna(raw_df['WeatherDelay'].mean())
```

In [33]:
```
raw_df['NASDelay']=raw_df['NASDelay'].fillna(raw_df['NASDelay'].mean())
```

In [34]:
```
raw_df['SecurityDelay']=raw_df['SecurityDelay'].fillna(raw_df['SecurityDelay'].mean())
```

In [35]:
```
raw_df['LateAircraftDelay']=raw_df['LateAircraftDelay'].fillna(raw_df['LateAircraftDelay'].mean())
```

In [36]:
```
raw_df.isna().sum()
```

Out[36]:
```
Unnamed: 0                0
Year                      0
Month                     0
DayofMonth                0
DayOfWeek                 0
DepTime                   0
CRSDepTime                0
ArrTime                7110
CRSArrTime                0
UniqueCarrier             0
FlightNum                 0
TailNum                   5
ActualElapsedTime      8387
CRSElapsedTime          198
AirTime                8387
ArrDelay                  0
DepDelay                  0
Origin                    0
Dest                      0
Distance                  0
TaxiIn                 7110
TaxiOut                 455
Cancelled                 0
CancellationCode          0
Diverted                  0
CarrierDelay              0
WeatherDelay              0
NASDelay                  0
SecurityDelay             0
LateAircraftDelay         0
dtype: int64
```

Tras comprobar que se han reemplazado los datos correctamente, se decide eliminar el resto de NaN

In [37]:
```
no_NaN_df = raw_df.dropna()
```

In [38]:
```
no_NaN_df.isna().sum()
```

Out[38]:
```
Unnamed: 0           0
Year                 0
Month                0
DayofMonth           0
DayOfWeek            0
DepTime              0
CRSDepTime           0
ArrTime              0
CRSArrTime           0
UniqueCarrier        0
FlightNum            0
TailNum              0
ActualElapsedTime    0
CRSElapsedTime       0
AirTime              0
ArrDelay             0
DepDelay             0
Origin               0
Dest                 0
Distance             0
TaxiIn               0
TaxiOut              0
Cancelled            0
CancellationCode     0
Diverted             0
CarrierDelay         0
WeatherDelay         0
NASDelay             0
SecurityDelay        0
LateAircraftDelay    0
dtype: int64
```

Se convierten los datos categóricos usando el método Dummies.

In [39]:
```
dummies_df = pd.get_dummies(no_NaN_df, columns=["UniqueCarrier", "Origin", "Dest"])
```

In [40]:
```
dummies_df
```

Out[40]:

	Unnamed: 0	Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	FlightNum	TailNum	Acti
0	0	2008	1	3	4	2003.0	1955	2211.0	2225	335	N712SW	
1	1	2008	1	3	4	754.0	735	1002.0	1000	3231	N772SW	
2	2	2008	1	3	4	628.0	620	804.0	750	448	N428WN	
3	3	2008	1	3	4	1829.0	1755	1959.0	1925	3920	N464WN	
4	5	2008	1	3	4	1940.0	1915	2121.0	2110	378	N726SW	
...	...	...	...	...	...	...	...	...	...	...	...	...
1936753	7009710	2008	12	13	6	1250.0	1220	1617.0	1552	1621	N938DL	
1936754	7009717	2008	12	13	6	657.0	600	904.0	749	1631	N3743H	
1936755	7009718	2008	12	13	6	1007.0	847	1149.0	1010	1631	N909DA	
1936756	7009726	2008	12	13	6	1251.0	1240	1446.0	1437	1639	N646DL	
1936757	7009727	2008	12	13	6	1110.0	1103	1413.0	1418	1641	N908DL	

1928368 rows × 652 columns

Nivell 1

- **Exercici 1:**

Partein el conjunt de dadesDelayedFlights.csv en train i test. Estudia els dos conjunts per separat, a nivell descriptiu.

Finalmente, se aplica la función train_test_split() controlando el tamaño y random_state para usar siempre los mismos datos.

In [41]:
```
x = dummies_df[['DepTime', 'CRSDepTime']]
y = dummies_df[['ArrTime', 'CRSArrTime']]
```

In [42]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.33, random_state = 42)
```

In [43]:
```
X_train.shape
```

Out[43]:
```
(1292006, 2)
```

In [44]:
```
X_train
```

Out[44]:

	DepTime	CRSDepTime
1415709	744.0	730
624771	1937.0	1930
1588707	1304.0	1241
283507	1219.0	1150
646115	1947.0	1900
...	...	...
260236	1919.0	1900
1419937	2047.0	2005
132332	936.0	920
673666	721.0	710
122329	1021.0	940

1292006 rows × 2 columns

In [45]:
```
X_test.shape
```

Out[45]:
```
(636362, 2)
```

In [46]:
```
X_test
```

Out[46]:

	DepTime	CRSDepTime
660000	1705.0	1656
367101	1447.0	1415
451649	1850.0	1829
175660	1106.0	935
575555	1840.0	1820
...	...	...
95539	1335.0	1315
1133037	1910.0	1630
1156881	858.0	842
274863	1805.0	1745
1509604	1534.0	1520

636362 rows × 2 columns

In [47]:
```
y_train.shape
```

Out[47]:
```
(1292006, 2)
```

In [48]:
```
y_train
```

Out[48]:

	ArrTime	CRSArrTime
1415709	1610.0	1555
624771	5.0	14
1588707	1514.0	1458
283507	1956.0	1921
646115	2112.0	2029
...	...	...
260236	2151.0	2146
1419937	43.0	2243
132332	1133.0	1109
673666	829.0	815
122329	1043.0	1000

1292006 rows × 2 columns

In [49]:
```
y_test.shape
```

Out[49]:
```
(636362, 2)
```

In [50]:
```
y_test
```

Out[50]:

	ArrTime	CRSArrTime
660000	2000.0	1945
367101	1814.0	1805
451649	13.0	2359
175660	1351.0	1230
575555	1954.0	1935
...	...	...
95539	1340.0	1330
1133037	2112.0	1852
1156881	1044.0	1008
274863	2003.0	1954
1509604	1712.0	1725

636362 rows × 2 columns