In [ ]:

```
#S10 T01: Aprenentatge Supervisat
```

In [ ]:

```
#Nivell 1
```

In [ ]:

```
#Exercici 1
#Crea almenys tres models de classificació diferents per intentar predir el millor poss
ible l'endarreriment
#dels vols (ArrDelay) de DelayedFlights.csv. Considera si el vol ha arribat tard o no
 (ArrDelay > 0).
```

In [1]:

```
#Llibreries import
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [2]:

```
#Llibreries from
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_boston
from sklearn import metrics
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_digits
from sklearn.model_selection import cross_validate
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

In [3]:

```
#Dataset
df_dades=pd.read_csv('Python\DelayedFlights.csv', engine="python", error_bad_lines=Fals
e, warn_bad_lines=False, sep=',')
```

In [4]:

```
# Netegem el dataset d'atributs innecesaris.
df_dades.drop(['Year','Month','DayofMonth','DayOfWeek','TailNum','Cancelled', 'Cancella
tionCode', 'Diverted', 'UniqueCarrier', 'Origin', 'Dest'], axis=1, inplace=True)
```

In [5]:

```
# Suprimim valors no numerics.
df_dades=df_dades.dropna()
```

In [6]:

```
# Utilitzarem només una petita part de les instàncies del dataframe.
df_dades=df_dades.head(50)
```

In [235]:

```
df_dades
```

Out[235]:

	Unnamed: 0	DepTime	CRSDepTime	ArrTime	CRSArrTime	FlightNum	ActualElapsedTime
3	4	1829.0	1755	1959.0	1925	3920	90.0
5	6	1937.0	1830	2037.0	1940	509	240.0
7	11	1644.0	1510	1845.0	1725	1333	121.0
9	16	1452.0	1425	1640.0	1625	675	228.0
11	18	1323.0	1255	1526.0	1510	4	123.0
12	19	1416.0	1325	1512.0	1435	54	56.0
13	21	1657.0	1625	1754.0	1735	623	57.0
17	26	1422.0	1255	1657.0	1610	188	155.0
19	30	2107.0	1945	2334.0	2230	362	147.0
23	37	1812.0	1650	1927.0	1815	422	135.0
26	40	1326.0	1230	1559.0	1530	1056	153.0
32	48	1450.0	1435	1806.0	1745	3244	136.0
33	49	2245.0	1730	2354.0	1850	186	69.0
34	52	2025.0	1940	2135.0	2100	3154	70.0
35	53	1038.0	945	1314.0	1225	1035	96.0
36	54	1900.0	1850	2123.0	2045	205	143.0
37	56	948.0	925	959.0	940	3430	71.0
38	57	646.0	620	725.0	655	1580	99.0
39	58	1110.0	1040	1136.0	1110	2195	86.0
42	70	1813.0	1735	1936.0	1905	54	143.0
44	74	1734.0	1650	1941.0	1905	23	127.0
47	77	958.0	900	1052.0	950	1574	54.0
49	79	1538.0	1445	1753.0	1710	500	75.0
50	81	2248.0	2125	102.0	2345	890	74.0
51	82	1327.0	1230	1550.0	1500	1171	83.0
53	86	1832.0	1655	148.0	30	302	256.0
54	87	1229.0	1155	1633.0	1555	1079	124.0
56	89	2118.0	2015	144.0	50	2021	146.0
58	91	1739.0	1640	114.0	25	1018	275.0
59	94	1325.0	1240	1841.0	1810	419	196.0
60	95	1506.0	1440	2030.0	2010	2032	204.0
61	96	2039.0	1930	155.0	55	3940	196.0
62	98	1611.0	1535	1849.0	1825	538	98.0
63	99	1824.0	1715	117.0	25	2383	233.0
64	101	2118.0	2015	2224.0	2115	219	66.0
65	102	1818.0	1740	1916.0	1840	391	58.0

	Unnamed: 0	DepTime	CRSDepTime	ArrTime	CRSArrTime	FlightNum	ActualElapsedTime
66	104	2146.0	2055	2250.0	2155	815	64.0
67	105	2241.0	1910	2340.0	2010	1072	59.0
71	109	1726.0	1630	1832.0	1740	2284	66.0
72	111	1229.0	1220	1342.0	1325	2874	73.0
73	113	908.0	845	1628.0	1610	1774	260.0
74	114	1817.0	1730	122.0	50	2632	245.0
76	117	1849.0	1740	121.0	30	564	212.0
77	118	1210.0	1200	1905.0	1850	991	235.0
79	121	2232.0	2115	108.0	5	632	96.0
80	122	1512.0	1315	1802.0	1610	706	110.0
81	123	2025.0	1955	2301.0	2245	908	96.0
83	125	1627.0	1600	1916.0	1900	2907	109.0
84	126	1745.0	1710	2017.0	1945	2192	92.0
87	129	1725.0	1620	1940.0	1850	2803	135.0

In [236]:

```
#Correlació de dades.
df_dades.corr().round(3)
```

Out[236]:

	Unnamed: 0	DepTime	CRSDepTime	ArrTime	CRSArrTime	FlightNum	Ac
Unnamed: 0	1.000	0.130	0.154	-0.189	-0.191	0.174	
DepTime	0.130	1.000	0.980	0.037	0.093	-0.136	
CRSDepTime	0.154	0.980	1.000	0.014	0.092	-0.080	
ArrTime	-0.189	0.037	0.014	1.000	0.890	-0.202	
CRSArrTime	-0.191	0.093	0.092	0.890	1.000	-0.210	
FlightNum	0.174	-0.136	-0.080	-0.202	-0.210	1.000	
ActualElapsedTime	0.072	-0.045	-0.027	-0.364	-0.414	-0.010	
CRSElapsedTime	0.044	-0.025	-0.015	-0.376	-0.426	-0.034	
AirTime	0.029	-0.033	-0.020	-0.372	-0.422	-0.011	
ArrDelay	-0.004	0.444	0.267	0.117	0.045	-0.222	
DepDelay	-0.037	0.455	0.274	0.071	-0.001	-0.249	
Distance	0.124	-0.025	-0.014	-0.451	-0.490	0.024	
TaxiIn	-0.098	0.009	0.038	0.106	0.098	-0.154	
TaxiOut	0.616	-0.159	-0.122	-0.025	-0.042	0.168	
CarrierDelay	-0.013	0.273	0.111	0.188	0.080	-0.125	
WeatherDelay	NaN	NaN	NaN	NaN	NaN	NaN	
NASDelay	0.074	0.100	0.143	-0.081	-0.108	0.128	
SecurityDelay	-0.069	-0.195	-0.201	-0.070	-0.070	0.114	
LateAircraftDelay	0.003	0.370	0.299	-0.072	-0.023	-0.241	

In [237]:

```
#Assignació de dataset
X=df_dades #dataset
y=df_dades['ArrDelay']
```

In [238]:

```
#1.Model de classificació: The K-nearest neighbors
```

In [239]:

```
#Afegim un nou atribut al dataframe on posem valors True o False si el vol ha arribat t
ard o no (ArrDelay > 0).

for item in df_dades ['ArrDelay']:
    if item>0:
        df_dades['Delayed']=True
    else:
        df_dades['Delayed']=False
```

In [240]:

```
df_dades[['Delayed','FlightNum']]
```

Out[240]:

	Delayed	FlightNum
3	True	3920
5	True	509
7	True	1333
9	True	675
11	True	4
12	True	54
13	True	623
17	True	188
19	True	362
23	True	422
26	True	1056
32	True	3244
33	True	186
34	True	3154
35	True	1035
36	True	205
37	True	3430
38	True	1580
39	True	2195
42	True	54
44	True	23
47	True	1574
49	True	500
50	True	890
51	True	1171
53	True	302
54	True	1079
56	True	2021
58	True	1018
59	True	419
60	True	2032
61	True	3940
62	True	538
63	True	2383
64	True	219
65	True	391
66	True	815

	Delayed	FlightNum
67	True	1072
71	True	2284
72	True	2874
73	True	1774
74	True	2632
76	True	564
77	True	991
79	True	632
80	True	706
81	True	908
83	True	2907
84	True	2192
87	True	2803

In [241]:

```
#Divisió de les dades
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
```

In [242]:

```
#Escalat de les dades
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

In [243]:

```
classifier = KNeighborsClassifier(n_neighbors=4)
classifier.fit(X_train, y_train)
```

Out[243]:

```
KNeighborsClassifier(n_neighbors=4)
```

In [244]:

```
y_pred = classifier.predict(X_test)
```

In [266]:

```
#Matriu de confusió
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

Out[266]:

```
array([[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
        0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
        0, 0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
        0, 0]], dtype=int64)
```

In [268]:

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.0
```

In [269]:

```
print("Precision:",metrics.precision_score(y_test, y_pred, average='weighted',zero_division=0))
```

```
Precision: 0.0
```

In [270]:

```
print("Recall:",metrics.recall_score(y_test, y_pred, average='macro',zero_division=0))
```

```
Recall: 0.0
```

In [271]:

```
#2.Model de classificació: Support vector machine
```

In [272]:

```
X=df_dades #dataset
y=df_dades['ArrDelay']
```

In [273]:

```
#Divisió del dataset en train i test.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
```

In [274]:

```
clf = svm.SVC(kernel='linear') # Linear Kernel
```

In [275]:

```
clf.fit(X_train, y_train)
```

Out[275]:

```
SVC(kernel='linear')
```

In [276]:

```
y_pred = clf.predict(X_test)
```

In [277]:

```python
#Matriu de confusió
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

Out[277]:

```
array([[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 1],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0]], dtype=int64)
```

In [278]:

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.0
```

In [279]:

```
print("Precision:",metrics.precision_score(y_test, y_pred, average='weighted', zero_division=0))
```

```
Precision: 0.0
```

In [280]:

```
print("Recall:",metrics.recall_score(y_test, y_pred, average='macro', zero_division=0))
```

```
Recall: 0.0
```

In [281]:

```
#3.Model de classificació: Regressió logística I
```

In [282]:

```
X=df_dades #dataset
y=df_dades['ArrDelay']
```

In [283]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.33,random_state=0)
```

In [289]:

```
logreg = LogisticRegression(max_iter=2000)
```

In [290]:

```
logreg.fit(X_train,y_train)
```

Out[290]:

```
LogisticRegression(max_iter=2000)
```

In [291]:

```
y_pred=logreg.predict(X_test)
```

In [292]:

```
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

Out[292]:

```
array([[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1],
       [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
        0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0]], dtype=int64)
```

In [294]:

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.0
```

In [295]:

```
print("Precision:",metrics.precision_score(y_test, y_pred, average='weighted', zero_division=0))
```

```
Precision: 0.0
```

In [296]:

```
print("Recall:",metrics.recall_score(y_test, y_pred, average='macro', zero_division=0))
```

```
Recall: 0.0
```

In [297]:

```
#4.Model de classificació: Regressió logística II
```

In [298]:

```
X=df_dades #dataset
y=df_dades['ArrDelay']
```

In [299]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.33,random_state=0)
```

In [304]:

```
logisticRegr = LogisticRegression(max_iter=2000)
```

In [305]:

```
logisticRegr.fit(X_train, y_train)
```

Out[305]:

```
LogisticRegression(max_iter=2000)
```

In [306]:

```
y_pred = logisticRegr.predict(X_test)
```

In [307]:

```
cm = metrics.confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
```

In [308]:

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.0
```

In [309]:

```
print("Precision:",metrics.precision_score(y_test, y_pred, average='weighted', zero_div
ision=0))
```

```
Precision: 0.0
```

In [310]:

```
print("Recall:",metrics.recall_score(y_test, y_pred, average='macro', zero_division=0))
```

```
Recall: 0.0
```

In [227]:

```
#Exercici 2
#Compara els models de classificació utilitzant la precisió (accuracy), una matriu de c
onfusió i d'altres mètriques més avançades.
```

In [ ]:

```
#Accuracy
```

In [ ]:

```
#The K-nearest neighbors: 0.0
#Support vector machine: 0.0
#Regressió logística I: 0.0
#Regressió logística II: 0.0
```

In [ ]:

```
#Precision
```

In [ ]:

```
#The K-nearest neighbors: 0.0
#Support vector machine: 0.0
#Regressió logística I: 0.0
#Regressió logística II: 0.0
```

In [ ]:

```
#Recall
```

In [ ]:

```
#The K-nearest neighbors: 0.0
#Support vector machine: 0.0
#Regressió logística I: 0.0
#Regressió logística II: 0.0
```

In [ ]:

```
#Els valors de Accuracy, Precision i Recall coincideixen en valors números de 0.0
#Degut al volum alt de dades contingut al dataframe l'execució provocava un enorme rale
ntiment de l'execució del notebook fins al punt de
#reiniciar la màquina per no donar resposta al manegament de l'eina Jupyter.
#S'ha procedit a reduir al minim el conjunt de dades existents al dataframe. Tot reduin
t el volum del dataframe l'execució del notebook
#ha funcionat de manera positiva(després d'afinar la parametrització de les sentencie
s).
#Entenc els resultats obtinguts (Accuracy, Precision, Recall) al reduit nombre de dades
emprades en aquest exercici.
```

In [ ]:

```
#Exercici 3
#Entrena'ls utilitzant els diferents paràmetres que admeten.
```

In [ ]:

```
#Model A: The K-nearest neighbors
```

In [174]:

```
#Llibreries import
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [175]:

```
#Llibreries from
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_boston
from sklearn import metrics
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_digits
from sklearn.model_selection import cross_validate
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

In [180]:

```
#Escollim els atributs AirTime i Distance
X=df_dades[['AirTime','Distance']].dropna()
y=df_dades['ArrDelay'].dropna()  #atribut per realitzar la predicció
```

In [181]:

```
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.2, random_state=24)
```

In [182]:

```
train_A=KNeighborsClassifier(n_neighbors=4)
train_A.fit(X_train, y_train)
#Entrenament del model The K-nearest neighbors
predict=train_A.predict(X_test)
```

In [183]:

```
cnf_matrix_A = metrics.confusion_matrix(y_test, predict_A)
cnf_matrix_A
```

Out[183]:

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [185]:

```
print("Accuracy:",metrics.accuracy_score(y_test, predict_A))
```

```
Accuracy: 0.014953004472677773
```

In [186]:

```
print("Precision:",metrics.precision_score(y_test, predict_A, average='weighted', zero_
division=0))
```

```
Precision: 0.01126389752081318
```

In [187]:

```
print("Recall:",metrics.recall_score(y_test, predict_A, average='macro', zero_division=
0))
```

```
Recall: 0.0017027199969205202
```

In [ ]:

```
#Model B: Support vector machine
```

In [ ]:

```
#Llibreries import
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [ ]:

```
#Llibreries from
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_boston
from sklearn import metrics
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_digits
from sklearn.model_selection import cross_validate
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

In [7]:

```
#Escollim els atributs AirTime i Distance
X=df_dades[['AirTime','Distance']].dropna()
y=df_dades['ArrDelay'].dropna() #atribut per realitzar la predicció
```

In [8]:

```
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.2, random_state=24)
```

In [9]:

```
train_B=svm.SVC(kernel='linear') # Linear Kernel
#Entrenament del model Support vector machine
train_B.fit(X_train, y_train)
predict_B = train_B.predict(X_test)
```

In [10]:

```
cnf_matrix_B = metrics.confusion_matrix(y_test, predict_B)
cnf_matrix_B
```

Out[10]:

```
array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]], dtype=int64)
```

In [11]:

```
print("Accuracy:",metrics.accuracy_score(y_test, predict_B))
```

```
Accuracy: 0.0
```

In [12]:

```
print("Precision:",metrics.precision_score(y_test, predict_B, average='weighted', zero_
division=0))
```

```
Precision: 0.0
```

In [13]:

```
print("Recall:",metrics.recall_score(y_test, predict_B, average='macro', zero_division=
0))
```

```
Recall: 0.0
```

In [ ]:

```
#Model C: Regressió logística I
```

In [173]:

```
#Llibreries import
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [ ]:

```
#Llibreries from
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_boston
from sklearn import metrics
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_digits
from sklearn.model_selection import cross_validate
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

In [ ]:

```
#Escollim els atributs AirTime i Distance
X=df_dades[['AirTime','Distance']].dropna()
y=df_dades['ArrDelay'].dropna() #atribut per realitzar la predicció
```

In [14]:

```
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.2, random_state=24)
```

In [52]:

```
train_C=LogisticRegression(max_iter=2000)
#Entrenament del model Regressió logística I
train_C.fit(X_train,y_train)
predict_C=train_C.predict(X_test)
```

In [53]:

```
cnf_matrix_C = metrics.confusion_matrix(y_test, predict_C)
cnf_matrix_C
```

Out[53]:

```
array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]], dtype=int64)
```

In [54]:

```
print("Accuracy:",metrics.accuracy_score(y_test, predict_C))
```

```
Accuracy: 0.0
```

In [55]:

```
print("Precision:",metrics.precision_score(y_test, predict_C, average='weighted', zero_
division=0))
```

```
Precision: 0.0
```

In [56]:

```
print("Recall:",metrics.recall_score(y_test, predict_C, average='macro', zero_division=
0))
```

```
Recall: 0.0
```

In [ ]:

```
#Model D: Regressió logística II
```

In [ ]:

```
#Llibreries import
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [ ]:

```
#Llibreries from
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_boston
from sklearn import metrics
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_digits
from sklearn.model_selection import cross_validate
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

In [27]:

```
#Escollim els atributs AirTime i Distance
X=df_dades[['AirTime','Distance']].dropna()
y=df_dades['ArrDelay'].dropna() #atribut per realitzar la predicció
```

In [28]:

```
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.2, random_state=24)
```

In [57]:

```
train_D = LogisticRegression(max_iter=2000)
#Entrenament del model Regressió logística II
train_D.fit(X_train, y_train)
predict_D = train_D.predict(X_test)
```

In [58]:

```
cnf_matrix_D = metrics.confusion_matrix(y_test, predict_D)
cnf_matrix_D
```

Out[58]:

```
array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]], dtype=int64)
```

In [59]:

```
print("Accuracy:",metrics.accuracy_score(y_test, predict_D))
```

```
Accuracy: 0.0
```

In [60]:

```
print("Precision:",metrics.precision_score(y_test, predict_D, average='weighted', zero_
division=0))
```

```
Precision: 0.0
```

In [61]:

```
print("Recall:",metrics.recall_score(y_test, predict_D, average='macro', zero_division=
0))
```

```
Recall: 0.0
```

In [ ]:

```
#Exercici 4
#Compara el seu rendiment utilitzant l'aproximació traint/test o cross-validation.
```

In [123]:

```
#Llibreries import
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [124]:

```
#Llibreries from
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_boston
from sklearn import metrics
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_digits
from sklearn.model_selection import cross_validate
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.model_selection import cross_validate
from sklearn.model_selection import cross_val_score
from sklearn.metrics import recall_score
from sklearn.model_selection import KFold
```

In [126]:

```
#Escollim tots els atributs
X=df_dades[['Distance','TaxiIn','TaxiOut','CarrierDelay','WeatherDelay','NASDelay','Sec
urityDelay','LateAircraftDelay']].dropna()
y=df_dades['ArrDelay'].dropna() #atribut per realitzar la predicció
```

In [127]:

```
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.2, random_state=0)
```

In [129]:

```
clf=svm.SVC(kernel='linear', C=1, random_state=42)
kfold=KFold(n_splits=2, random_state=None)
scores=cross_val_score(clf, X, y, cv=kfold)
```

In [132]:

```
print("Cross Validation Scores:", scores)
```

```
Cross Validation Scores: [0. 0.]
```

In [ ]:

```
#Nivell 2
```

In [ ]:

```
#Exercici 5
#Realitza algun procés d'enginyeria de variables per millorar-ne la predicció
```

In [152]:

```
df_dades.dtypes
```

Out[152]:

```
Unnamed: 0            int64
Year                  int64
Month                 int64
DayofMonth            int64
DayOfWeek             int64
DepTime             float64
CRSDepTime            int64
ArrTime             float64
CRSArrTime            int64
UniqueCarrier        object
FlightNum             int64
TailNum              object
ActualElapsedTime   float64
CRSElapsedTime      float64
AirTime             float64
ArrDelay            float64
DepDelay            float64
Origin               object
Dest                 object
Distance              int64
TaxiIn              float64
TaxiOut             float64
Cancelled             int64
CancellationCode     object
Diverted              int64
CarrierDelay        float64
WeatherDelay        float64
NASDelay            float64
SecurityDelay       float64
LateAircraftDelay   float64
dtype: object
```

In [153]:

```
#Estandarització d'atributs numerics de tipus int.
df_dades[['CRSDepTime','CRSArrTime','FlightNum','Distance']]
```

Out[153]:

	CRSDepTime	CRSArrTime	FlightNum	Distance
0	1955	2225	335	810
1	735	1000	3231	810
2	620	750	448	515
3	1755	1925	3920	515
4	1915	2110	378	688
...	...	...	...	...
1936753	1220	1552	1621	906
1936754	600	749	1631	481
1936755	847	1010	1631	689
1936756	1240	1437	1639	533
1936757	1103	1418	1641	874

1936758 rows × 4 columns

In [154]:

```
#StandardScaler per valors números del dataframe df_dades
from sklearn.preprocessing import StandardScaler
#Implementation
data = df_dades[['CRSDepTime','CRSArrTime','FlightNum','Distance']]
data = data.dropna() #Esborrem valors NaN
scaler = StandardScaler() #
scaler.fit(data) #
print('<-- Dataframe wcupmatch_df preprocessat --> \n', scaler.transform(data))
```

```
<-- Dataframe wcupmatch_df preprocessat -->
 [[ 1.14775327  1.27148381 -0.95092383  0.07713737]
 [-1.72441169 -1.36499661  0.53825054  0.07713737]
 [-1.99514855 -1.90305383 -0.89281723 -0.43637094]
 ...
 [-1.46073753 -1.34347432 -0.28449773 -0.13348807]
 [-0.53552373 -0.42447257 -0.28038399 -0.40503823]
 [-0.85805373 -0.46536492 -0.27935556  0.18854257]]
```

In [155]:

```
scaled_data = scaler.transform(data)
print(scaled_data.mean(axis = 0))
```

```
[1.47159791e-16 1.69377573e-16 8.00661965e-17 4.03852956e-17]
```

In [156]:

```
print('<-- Dataframe wcupmatch_df preprocessat --> \n', scaler.mean_)
```

```
<-- Dataframe wcupmatch_df preprocessat -->
 [1467.47264398 1634.22464087 2184.26323629  765.68615903]
```

In [157]:

```
#Standardització d'atributs numerics de tipus float.
df_dades[['DepTime','ArrTime','ActualElapsedTime','CRSElapsedTime','AirTime','ArrDelay'
,'TaxiIn','TaxiOut','CarrierDelay','WeatherDelay','NASDelay','SecurityDelay','LateAircr
aftDelay']]
```

Out[157]:

	DepTime	ArrTime	ActualElapsedTime	CRSElapsedTime	AirTime	ArrDelay	TaxiIn
0	2003.0	2211.0	128.0	150.0	116.0	-14.0	4.0
1	754.0	1002.0	128.0	145.0	113.0	2.0	5.0
2	628.0	804.0	96.0	90.0	76.0	14.0	3.0
3	1829.0	1959.0	90.0	90.0	77.0	34.0	3.0
4	1940.0	2121.0	101.0	115.0	87.0	11.0	4.0
...	...	...	...	...	...	...	...
1936753	1250.0	1617.0	147.0	152.0	120.0	25.0	9.0
1936754	657.0	904.0	127.0	109.0	78.0	75.0	15.0
1936755	1007.0	1149.0	162.0	143.0	122.0	99.0	8.0
1936756	1251.0	1446.0	115.0	117.0	89.0	9.0	13.0
1936757	1110.0	1413.0	123.0	135.0	104.0	-5.0	8.0

1936758 rows × 13 columns

In [158]:

```
#StandardScaler per valors float del dataframe df_dades
from sklearn.preprocessing import StandardScaler
#Implementation
data = df_dades[['DepTime','ArrTime','ActualElapsedTime','CRSElapsedTime','AirTime','ArrDelay','TaxiIn','TaxiOut','CarrierDelay','WeatherDelay','NASDelay','SecurityDelay','LateAircraftDelay']]
data = data.dropna() #Esborrem valors NaN
scaler = StandardScaler() #
scaler.fit(data) #
print('<-- Dataframe wcupmatch_df preprocessat --> \n', scaler.transform(data))
```

```
<-- Dataframe wcupmatch_df preprocessat -->
 [[ 5.94651689e-01  5.86104520e-01 -6.27665526e-01 ... -4.43993094e-01
  -4.45624906e-02  1.59399800e-01]
 [ 8.32364614e-01  7.19679633e-01  1.44713010e+00 ... -4.43993094e-01
  -4.45624906e-02  5.16076899e-01]
 [ 1.87458253e-01  3.90879354e-01 -1.98874430e-01 ... -4.43993094e-01
  -4.45624906e-02  1.11053873e+00]
 ...
 [-6.79753714e-01  4.29023148e-04  1.60756811e-01 ... -4.43993094e-01
  -4.45624906e-02 -7.83849329e-02]
 [-1.98497376e+00 -1.22058451e+00 -1.15882605e-01 ...  8.80312453e-02
  -4.45624906e-02 -6.01511345e-01]
 [-1.21460780e+00 -8.01021656e-01  3.68236373e-01 ...  1.17588153e-01
  -4.45624906e-02  1.27698804e+00]]
```

In [159]:

```
scaled_data = scaler.transform(data)
print(scaled_data.mean(axis = 0))
```

```
[ 1.67000508e-16 -5.94184619e-17  1.26856593e-16 -1.62443878e-16
  5.51807955e-17 -2.26008873e-17  1.85910525e-17 -1.19839383e-17
 -2.55399140e-17  1.76797264e-17 -5.22018983e-17 -4.46549790e-18
  5.17633226e-17]
```

In [160]:

```
print('<-- Dataframe wcupmatch_df preprocessat --> \n', scaler.mean_)
```

```
<-- Dataframe wcupmatch_df preprocessat -->
 [1.55883218e+03 1.61674948e+03 1.35377881e+02 1.31764029e+02
 1.07420290e+02 6.32912084e+01 7.29723572e+00 2.06603550e+01
 1.91793989e+01 3.70357070e+00 1.50216355e+01 9.01371396e-02
 2.52964662e+01]
```

In [161]:

```
#Depuració d'outliers.
```

In [162]:

```
#L'atribut WeatherDelay té valors de 1000 o superiors. Interpretem que es tracta de val
ors outliers
print(data['WeatherDelay'].head(50)>=1000)
```

```
3     False
5     False
7     False
9     False
11    False
12    False
13    False
17    False
19    False
23    False
26    False
32    False
33    False
34    False
35    False
36    False
37    False
38    False
39    False
42    False
44    False
47    False
49    False
50    False
51    False
53    False
54    False
56    False
58    False
59    False
60    False
61    False
62    False
63    False
64    False
65    False
66    False
67    False
71    False
72    False
73    False
74    False
76    False
77    False
79    False
80    False
81    False
83    False
84    False
87    False
Name: WeatherDelay, dtype: bool
```

In [163]:

```
#Esborrem les instàncies que hem indicat com a outliers
data.drop(data[data.WeatherDelay >= 1000].index, inplace=True)
```

In [164]:

```
data['WeatherDelay']
```

Out[164]:

```
3           0.0
5           0.0
7           0.0
9           0.0
11          0.0
           ...
1936751     0.0
1936752     0.0
1936753     0.0
1936754    57.0
1936755     0.0
Name: WeatherDelay, Length: 1247476, dtype: float64
```

In [165]:

```
data
```

Out[165]:

	DepTime	ArrTime	ActualElapsedTime	CRSElapsedTime	AirTime	ArrDelay	TaxiIn
3	1829.0	1959.0	90.0	90.0	77.0	34.0	3.0
5	1937.0	2037.0	240.0	250.0	230.0	57.0	3.0
7	1644.0	1845.0	121.0	135.0	107.0	80.0	6.0
9	1452.0	1640.0	228.0	240.0	213.0	15.0	7.0
11	1323.0	1526.0	123.0	135.0	110.0	16.0	4.0
...	...	...	...	...	...	...	...
1936751	921.0	1112.0	111.0	98.0	82.0	64.0	8.0
1936752	1552.0	1735.0	43.0	58.0	27.0	17.0	9.0
1936753	1250.0	1617.0	147.0	152.0	120.0	25.0	9.0
1936754	657.0	904.0	127.0	109.0	78.0	75.0	15.0
1936755	1007.0	1149.0	162.0	143.0	122.0	99.0	8.0

1247476 rows × 13 columns

In [166]:

```
print(max(data['WeatherDelay']))#Ara el valor outlier per amunt és 996.0
```

```
996.0
```

In [167]:

```
#L'atribut WeatherDelay té valors de 0.0. Interpretem que es tracta de valors outliers.
print(data['WeatherDelay']==0.0)
```

```
3           True
5           True
7           True
9           True
11          True
           ...
1936751     True
1936752     True
1936753     True
1936754    False
1936755     True
Name: WeatherDelay, Length: 1247476, dtype: bool
```

In [168]:

```
#Esborrem les instàncies que hem indicat com a outliers
data.drop(data[data.WeatherDelay == 0.0].index, inplace=True)
```

In [170]:

```
print(min(data['WeatherDelay'], default="EMPTY"))##Ara el valor outlier per sota és 1.0
```

```
1.0
```

In [ ]:

```
#Nivell 3
```

In [ ]:

```
#Exercici 6
#No utilitzis la variable DepDelay a l'hora de fer prediccions
```

In [1]:

```
#Llibreries import
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [2]:

```
#Llibreries from
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_boston
from sklearn import metrics
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_digits
from sklearn.model_selection import cross_validate
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

In [3]:

```
#Dataset
df_dades=pd.read_csv('Python\DelayedFlights.csv', engine="python", error_bad_lines=Fals
e, warn_bad_lines=False, sep=',')
```

In [4]:

```
# Netegem el dataset d'atributs innecesaris.
df_dades.drop(['Year','Month','DayofMonth','DayOfWeek','TailNum','Cancelled', 'Cancella
tionCode', 'Diverted', 'UniqueCarrier', 'Origin', 'Dest'], axis=1, inplace=True)
```

In [5]:

```
# Netegem també el dataset de l'atribut DepDelay.
df_dades.drop(['DepDelay'], axis=1, inplace=True)
```

In [6]:

```
# Suprimim valors no numerics.
df_dades=df_dades.dropna()
```

In [7]:

```
# Utilitzarem només una petita part de les instàncies del dataframe.
df_dades=df_dades.head(50)
```

In [8]:

```
df_dades.corr().round(3)
```

Out[8]:

	Unnamed: 0	DepTime	CRSDepTime	ArrTime	CRSArrTime	FlightNum	Ac
Unnamed: 0	1.000	0.130	0.154	-0.189	-0.191	0.174	
DepTime	0.130	1.000	0.980	0.037	0.093	-0.136	
CRSDepTime	0.154	0.980	1.000	0.014	0.092	-0.080	
ArrTime	-0.189	0.037	0.014	1.000	0.890	-0.202	
CRSArrTime	-0.191	0.093	0.092	0.890	1.000	-0.210	
FlightNum	0.174	-0.136	-0.080	-0.202	-0.210	1.000	
ActualElapsedTime	0.072	-0.045	-0.027	-0.364	-0.414	-0.010	
CRSElapsedTime	0.044	-0.025	-0.015	-0.376	-0.426	-0.034	
AirTime	0.029	-0.033	-0.020	-0.372	-0.422	-0.011	
ArrDelay	-0.004	0.444	0.267	0.117	0.045	-0.222	
Distance	0.124	-0.025	-0.014	-0.451	-0.490	0.024	
TaxiIn	-0.098	0.009	0.038	0.106	0.098	-0.154	
TaxiOut	0.616	-0.159	-0.122	-0.025	-0.042	0.168	
CarrierDelay	-0.013	0.273	0.111	0.188	0.080	-0.125	
WeatherDelay	NaN	NaN	NaN	NaN	NaN	NaN	
NASDelay	0.074	0.100	0.143	-0.081	-0.108	0.128	
SecurityDelay	-0.069	-0.195	-0.201	-0.070	-0.070	0.114	
LateAircraftDelay	0.003	0.370	0.299	-0.072	-0.023	-0.241	

In [9]:

```
X=df_dades #dataset
y=df_dades['ArrDelay'] #Atribut per realitzar la predicció
```

In [10]:

```
#1.Model de classificació: The K-nearest neighbors
```

In [11]:

```
#Afegim un nou atribut al dataframe on posem valors True o False si el vol ha arribat t
ard o no (ArrDelay > 0).

for item in df_dades ['ArrDelay']:
    if item>0:
        df_dades['Delayed']=True
    else:
        df_dades['Delayed']=False
```

In [12]:

```
df_dades[['Delayed','FlightNum']]
```

Out[12]:

	Delayed	FlightNum
3	True	3920
5	True	509
7	True	1333
9	True	675
11	True	4
12	True	54
13	True	623
17	True	188
19	True	362
23	True	422
26	True	1056
32	True	3244
33	True	186
34	True	3154
35	True	1035
36	True	205
37	True	3430
38	True	1580
39	True	2195
42	True	54
44	True	23
47	True	1574
49	True	500
50	True	890
51	True	1171
53	True	302
54	True	1079
56	True	2021
58	True	1018
59	True	419
60	True	2032
61	True	3940
62	True	538
63	True	2383
64	True	219
65	True	391
66	True	815

	Delayed	FlightNum
67	True	1072
71	True	2284
72	True	2874
73	True	1774
74	True	2632
76	True	564
77	True	991
79	True	632
80	True	706
81	True	908
83	True	2907
84	True	2192
87	True	2803

In [13]:

```
#Divisió de les dades
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
```

In [14]:

```
#Escalat de les dades
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

In [15]:

```
classifier = KNeighborsClassifier(n_neighbors=4)
classifier.fit(X_train, y_train)
```

Out[15]:

```
KNeighborsClassifier(n_neighbors=4)
```

In [16]:

```
y_pred = classifier.predict(X_test)
```

In [17]:

```
#Matriu de confusió
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

Out[17]:

```
array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0]], dtype=int64)
```

In [102]:

```
print("Accuracy sense atribut DelDelay:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy sense atribut DelDelay: 0.0
```

In [103]:

```
print("Precision sense atribut DelDelay:",metrics.precision_score(y_test, y_pred, avera
ge='weighted',zero_division=0))
```

```
Precision sense atribut DelDelay: 0.0
```

In [104]:

```
print("Recall sense atribut DelDelay:",metrics.recall_score(y_test, y_pred, average='ma
cro',zero_division=0))
```

```
Recall sense atribut DelDelay: 0.0
```

In [22]:

```
#2.Model de classificació: Support vector machine
```

In [23]:

```
X=df_dades #dataset
y=df_dades['ArrDelay']
```

In [24]:

```
#Divisió del dataset en train i test.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
```

In [25]:

```
clf = svm.SVC(kernel='linear') # Linear Kernel
```

In [26]:

```
clf.fit(X_train, y_train)
```

Out[26]:

```
SVC(kernel='linear')
```

In [27]:

```
y_pred = clf.predict(X_test)
```

In [28]:

```
#Matriu de confusio
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

Out[28]:

```
array([[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        1],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
        0],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0]], dtype=int64)
```

In [99]:

```
print("Accuracy sense atribut DelDelay:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy sense atribut DelDelay: 0.0
```

In [100]:

```
print("Precision sense atribut DelDelay:",metrics.precision_score(y_test, y_pred, avera
ge='weighted', zero_division=0))
```

```
Precision sense atribut DelDelay: 0.0
```

In [101]:

```
print("Recall sense atribut DelDelay:",metrics.recall_score(y_test, y_pred, average='ma
cro', zero_division=0))
```

```
Recall sense atribut DelDelay: 0.0
```

In [32]:

```
#3.Model de classificació: Regressió logística I
```

In [33]:

```
X=df_dades #dataset
y=df_dades['ArrDelay']
```

In [34]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.33,random_state=0)
```

In [86]:

```
logreg = LogisticRegression(max_iter=10000)
```

In [87]:

```
logreg.fit(X_train,y_train)
```

Out[87]:

```
LogisticRegression(max_iter=10000)
```

In [88]:

```
y_pred=logreg.predict(X_test)
```

In [89]:

```
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

Out[89]:

```
array([[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1],
       [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
        0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0]], dtype=int64)
```

In [96]:

```
print("Accuracy sense atribut DelDelay:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy sense atribut DelDelay: 0.0
```

In [97]:

```
print("Precision sense atribut DelDelay:",metrics.precision_score(y_test, y_pred, avera
ge='weighted', zero_division=0))
```

```
Precision sense atribut DelDelay: 0.0
```

In [98]:

```
print("Recall sense atribut DelDelay:",metrics.recall_score(y_test, y_pred, average='ma
cro', zero_division=0))
```

```
Recall sense atribut DelDelay: 0.0
```

In [73]:

```
#4.Model de classificació: Regressió logística II
```

In [74]:

```
X=df_dades #dataset
y=df_dades['ArrDelay']
```

In [75]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.33,random_state=0)
```

In [79]:

```
logisticRegr = LogisticRegression(max_iter=10000)
```

In [80]:

```
logisticRegr.fit(X_train, y_train)
```

Out[80]:

```
LogisticRegression(max_iter=10000)
```

In [81]:

```
y_pred = logisticRegr.predict(X_test)
```

In [82]:

```
cm = metrics.confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
```

In [93]:

```
print("Accuracy sense atribut DelDelay:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy sense atribut DelDelay: 0.0
```

In [94]:

```
print("Precision sense atribut DelDelay:",metrics.precision_score(y_test, y_pred, avera
ge='weighted', zero_division=0))
```

```
Precision sense atribut DelDelay: 0.0
```

In [95]:

```
print("Recall sense atribut DelDelay:",metrics.recall_score(y_test, y_pred, average='ma
cro', zero_division=0))
```

```
Recall sense atribut DelDelay: 0.0
```

In [ ]:

```

```