Solución Problema DataScience Hackathon CaixaBank

Primero cargué los datos de entrenamiento y eliminé los valores NA para evitar problemas al momento del entrenamiento del modelo.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
data = pd.read_csv('train.csv');
data.head()
```

Out[2]:

	Date	Open	High	Low	Close	Adj Close	Volume	Target
0	1994-01-03	3615.199951	3654.699951	3581.000000	3654.500000	3654.496338	0.0	0
1	1994-01-04	3654.500000	3675.500000	3625.100098	3630.300049	3630.296387	0.0	1
2	1994-01-05	3625.199951	3625.199951	3583.399902	3621.199951	3621.196289	0.0	1
3	1994-01-06	NaN	NaN	NaN	NaN	NaN	NaN	0
4	1994-01-07	3621.199951	3644.399902	3598.699951	3636.399902	3636.396240	0.0	1

In [5]:
```
data = data.dropna(axis = 0, how = "any")
data.head()
```

Out[5]:

	Date	Open	High	Low	Close	Adj Close	Volume	Target
0	1994-01-03	3615.199951	3654.699951	3581.000000	3654.500000	3654.496338	0.0	0
1	1994-01-04	3654.500000	3675.500000	3625.100098	3630.300049	3630.296387	0.0	1
2	1994-01-05	3625.199951	3625.199951	3583.399902	3621.199951	3621.196289	0.0	1
4	1994-01-07	3621.199951	3644.399902	3598.699951	3636.399902	3636.396240	0.0	1
5	1994-01-10	3655.199951	3678.199951	3655.199951	3660.600098	3660.596436	0.0	1

Como la fecha estaba en formato str creé una función que transformaba la fecha en un número utilizando la librería datetime. El valor 0 se asigna a la fecha más antigua de los dos datasets (03/01/1994).

In [9]:
```
# Convertir fecha a formato numerico empezando desde 0
def date_to_int(data):
    data2 = data;
    data2["Date"] = pd.to_datetime(data["Date"])
    data2["Date"] = data2["Date"].map(dt.datetime.toordinal)
    d0 = 727931; # Valor correspondiente a 1994-01-03;
    data2["Date"] = np.array(data2["Date"]).reshape(-1,1) - d0*np.ones([len(data2),1])
    return data2
```

In [10]:
```
data = date_to_int(data)
```

Como modelo escogí un árbol de decisión debido a que fue el modelo que me generó mejores resultados. Utilicé la validación cruzada KFold para evaluar el rendimiento del modelo con el dataset de entrenamiento.

In [28]:
```
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.feature_selection import RFE
```

In [14]:
```
 tree = DecisionTreeClassifier(criterion = "entropy", min_samples_split = 10, random_state = 99)
```

In [22]:
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
```

In [23]:
```
cv = KFold(n_splits = 20, shuffle = True, random_state = 1)
```

In [27]:
```
tree.fit(X,Y)
scores = cross_val_score(tree, X, Y, scoring = "accuracy", cv = cv, n_jobs = 1)
scores.mean()
```

Finalmente leí el dataset de test, apliqué la función creada anteriormente para transformar la fecha a formato numérico y creé los archivos de predicción.

In [50]:
```
Y_pred = tree.predict(test[predictors])
```

Creacion de archivos

In [51]:
```
with open("prediction.csv", "w") as outfile1:
    for i in range(len(Y_pred)):
        outfile1.write(str(Y_pred[i]))
        outfile1.write("\n")
```

In [52]:
```
import json
```

In [53]:
```
Y_pred_str = []
for i in Y_pred:
    Y_pred_str.append(str(i))
Y_pred_str[0]
```

Out[53]:
```
'0'
```

In [49]:
```
with open('prediction.json', 'w') as outfile2:
    json.dump(Y_pred_str, outfile2)
```