

Não se pode utilizar o google colab por que para se ter acesso ao terminal, para instalação do pyspark, é necessário a assinatura da plataforma, não consegui de outra forma.

Exercício

- Importamos as bibliotecas

```
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.feature import VectorAssembler, StringIndexer, IndexToString
from pyspark.sql import SparkSession
```

DecisionTreeClassifier, classe que constrói classificador da árvore de decisão.
VectorAssembler transforma colunas em vetor, que os algoritmos de ML conseguem ler.
StringIndexer é utilizado para transformar valores categóricos em numéricos
IndexToString é utilizado para transformar valores numéricos em categóricos
Importa a classe SparkSession para criar uma sessão do Spark.

- Criaremos a sessão Spark.

```
spark = SparkSession.builder.appName("app_model").getOrCreate()
```

- Carregaremos o ficheiro stocks_2021.csv num dataframe.

```
df = spark.read.option("header", "true").csv("stocks_2021.csv")
```

- Filtraremos as linhas com o valor 'BA', na coluna 'ticker'

```
stock1 = 'BA'
df_stock1 = df.filter(df.ticker == stock1)
```

- Modificaremos o tipo de dados das colunas open, low e close para float.

```
df = df.withColumn('open', df.open.cast('float'))
df = df.withColumn('low', df.low.cast('float'))
df = df.withColumn('close', df.close.cast('float'))
```

- Criaremos uma coluna 'features', utilizando as colunas open, low e close através do VectorAssembler.

```
va = VectorAssembler(inputCols=['open', 'low', 'close'], outputCol='features')
va_df = va.transform(df)
```

Crio um vector usando VectorAssembler, que combina as colunas 'open', 'low' e 'close' em uma unica coluna e depois aplico a transformação VectorAssembler ao DataFrame, criando um novo DataFrame va_df com a coluna 'features'

- Converteremos a coluna 'ticker' em tipo numérico, utilizando o StringIndexer e chamar-lhe-emos 'label'.

```
indexer = StringIndexer(inputCol='ticker', outputCol='label')
indexer_model = indexer.fit(va_df)
indexed_df = indexer_model.transform(va_df)
```

transformo os valores em ticker em valores numéricos e os coloco em label e treino o modelo com os novos valores de label


```
testingModel.py > ...
1  from pyspark.ml.classification import DecisionTreeClassifier
2
3  from pyspark.ml.feature import VectorAssembler, StringIndexer, IndexToString
4
5  from pyspark.sql import SparkSession
6
7  spark = SparkSession.builder.appName("app_model").getOrCreate()
8
9  df = spark.read.option("header", "true").csv("stocks_2021.csv")
10
11  stock1 = 'BA'
12  df_stock1 = df.filter(df.ticker == stock1)
13
14  df = df.withColumn('open', df.open.cast('float'))
15  df = df.withColumn('low', df.low.cast('float'))
16  df = df.withColumn('close', df.close.cast('float'))
17
18  va = VectorAssembler(inputCols=['open', 'low', 'close'], outputCol='features')
19
20  va_df = va.transform(df)
21
22  indexer = StringIndexer(inputCol='ticker', outputCol='label')
23
24  indexer_model = indexer.fit(va_df)
25  indexed_df = indexer_model.transform(va_df)
26
27  dtc = DecisionTreeClassifier(featuresCol='features', labelCol='label')
28
29  (train, test) = indexed_df.randomSplit([0.2, 0.8])
30
31  dtc_model = dtc.fit(train)
32
33  pred = dtc_model.transform(test)
34
35  converter = IndexToString(inputCol="label", outputCol="ticker_pred", labels=indexer_model.labels)
36
37  pred_with_ticker = converter.transform(pred)
38
39  joined_df = pred_with_ticker.join(df_stock1, on='ticker', how='inner')
40
41  joined_df.show()
42
43
```