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 algoritimos 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 VectorAssempler, 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 valures numéricos e os coloco em label e treino o modelo com os novos valores de label - Converteremos os valores 'ticker', de volta ao seu estado original, com IndexToString.

 $converter = IndexToString(inputCol="label", outputCol="ticker\_pred", labels=indexer\_model.labels) \\ pred\_with\_ticker = converter.transform(pred)$ 

Criando transformador usando IndexToString para reverter a transformação feita com StringIndexer. Pego os dados da coluna 'label' e ponho na nova coluna 'ticker\_pred' Utilizo o conversor para aplicar a transformação de conversão dos valores numéricos da coluna 'label' de volta aos valores categóricos da coluna 'ticker'

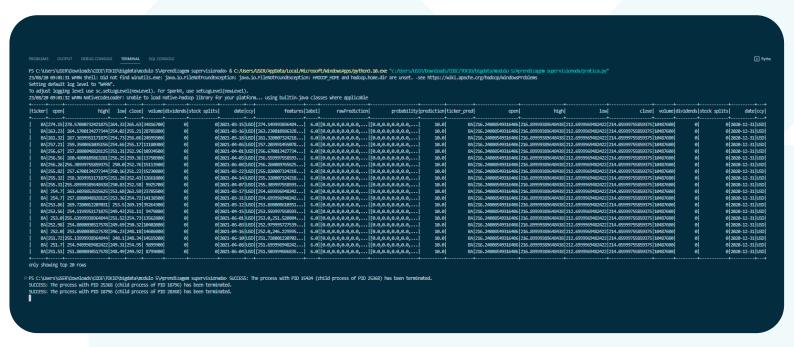
- Juntaremos, através de inner join, o dataframe com as previsões e o que contém os valores reajustados da coluna 'ticker'.

joined\_df = pred\_with\_ticker.join(df\_stock1, on='ticker', how='inner')

juntando o dataframe filtrado original df\_stock1 com o dataframe resultante da aplicação do modelo.

- Mostro os dados utilizando show()

joined df.show()



```
testingModel.py >
      from pyspark.ml.classification import DecisionTreeClassifier
     from pyspark.ml.feature import VectorAssembler, StringIndexer, IndexToString
     from pyspark.sql import SparkSession
     spark = SparkSession.builder.appName("app_model").getOrCreate()
     df = spark.read.option("header", "true").csv("stocks_2021.csv")
     stock1 = 'BA'
     df_stock1 = df.filter(df.ticker == stock1)
     df = df.withColumn('open', df.open.cast('float'))
     df = df.withColumn('low', df.low.cast('float'))
     df = df.withColumn('close', df.close.cast('float'))
     va = VectorAssembler(inputCols=['open', 'low', 'close'], outputCol='features')
     va_df = va.transform(df)
     indexer = StringIndexer(inputCol='ticker', outputCol='label')
      indexer_model = indexer.fit(va_df)
     indexed_df = indexer_model.transform(va_df)
     dtc = DecisionTreeClassifier(featuresCol='features', labelCol='label')
     (train, test) = indexed_df.randomSplit([0.2, 0.8])
     dtc_model = dtc.fit(train)
     pred = dtc_model.transform(test)
     converter = IndexToString(inputCol="label", outputCol="ticker_pred", labels=indexer_model.labels)
     pred_with_ticker = converter.transform(pred)
      joined_df = pred_with_ticker.join(df_stock1, on='ticker', how='inner')
      joined_df.show()
43
```