SPARK ML LABORATUAR MATERYALİ

1. Lab: Sınıflandırma
2. Lab: Pipeline oluşturma
3. Lab: Regresyon

Veri:

https://www.kaggle.com/datasets/tylerx/flights-and-airports-data

1.Lab : Sınıflandırma

Lab Adımları:

Sınıflandırma Modeli Oluşturma

Bu alıştırmada, uçuşun erteleneceğini öngörmek için bir uçuş özelliklerini kullanan bir sınıflandırma modeli uygulayacaksınız.

Spark SQL ve Spark ML Kütüphanelerini ekle

İlk önce ihtiyacınız olan kitaplıkları alın:

```
pip install pyspark
```

```
from pyspark.sql.types import *
from pyspark.sql.functions import *
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import VectorAssembler
```

Veriyi Yükleme

Bu alıştırma verileri, uçuşların ayrıntılarını içeren bir CSV dosyası olarak sağlanmaktadır. Veriler, her uçuşun belirli özelliklerini (veya özelliklerini) ve uçuşun kaç dakika önce geldiğini gösteren bir sütun içerir.

Bu verileri bir DataFrame'e yükleyip görüntüleyeceksiniz.

! Dosyanın bulunduğu dizini kontrol ediniz.

```
csv = spark.read.csv('/kaggle/input/flights-and-airports-data/flights.csv',
```

```
inferSchema=True, header=True)
csv.show()
```

```
+----------+---------+-------+---------------+-------------+--------+--------+
|DayofMonth|DayOfWeek|Carrier|OriginAirportID|DestAirportID|DepDelay|ArrDelay|
+----------+---------+-------+---------------+-------------+--------+--------+
|        19|        5|     DL|          11433|        13303|      -3|       1|
|        19|        5|     DL|          14869|        12478|       0|      -8|
|        19|        5|     DL|          14057|        14869|      -4|     -15|
|        19|        5|     DL|          15016|        11433|      28|      24|
|        19|        5|     DL|          11193|        12892|      -6|     -11|
|        19|        5|     DL|          10397|        15016|      -1|     -19|
|        19|        5|     DL|          15016|        10397|       0|      -1|
|        19|        5|     DL|          10397|        14869|      15|      24|
|        19|        5|     DL|          10397|        10423|      33|      34|
|        19|        5|     DL|          11278|        10397|     323|     322|
|        19|        5|     DL|          14107|        13487|      -7|     -13|
```

Verileri Hazırla

Çoğu modelleme, kapsamlı bir araştırma ve verilerin hazırlanmasıyla başlar. Bu örnekte, veriler sizin için temizlendi. Özellik olarak kullanılacak sütunların bir alt kümesini seçersiniz ve planlanan varış saatinden sonra 15 dakika veya daha fazla süratle gelen uçuşlar için **Late** adında **1** bir Boolean etiket alanı oluştursunuz, aksi takdirde uçuş erken veya zamanında **0** olur.

(Gerçek bir senaryoda, eksik veya çoğaltılmış verileri işleme, sayısal sütunları ölçeklendirme ve modeliniz için yeni özellikler oluşturmak için özellik mühendisliği adı verilen bir işlemi kullanma gibi ek görevleri yerine getireceğinizi unutmayın).

```
data = csv.select("DayofMonth", "DayOfWeek", "OriginAirportID", "DestAirportID",
"DepDelay", ((col("ArrDelay") > 15).cast("int").alias("Late")))
data.show()
```

```
+----------+---------+---------------+-------------+--------+----+
|DayofMonth|DayOfWeek|OriginAirportID|DestAirportID|DepDelay|Late|
+----------+---------+---------------+-------------+--------+----+
|        19|        5|          11433|        13303|      -3|   0|
|        19|        5|          14869|        12478|       0|   0|
|        19|        5|          14057|        14869|      -4|   0|
|        19|        5|          15016|        11433|      28|   1|
|        19|        5|          11193|        12892|      -6|   0|
|        19|        5|          10397|        15016|      -1|   0|
|        19|        5|          15016|        10397|       0|   0|
|        19|        5|          10397|        14869|      15|   1|
|        19|        5|          10397|        10423|      33|   1|
|        19|        5|          11278|        10397|     323|   1|
```

Verileri Böl

Kaynak verileri ayırmak için güdümlü(supervised) makine öğrenme modelleri oluştururken bazılarını modeli eğitmek ve bazılarını eğitilmiş modeli test etmek için ayırmak yaygın bir uygulamadır. Bu alıştırmada, eğitim için verilerin% 70'ini kullanacaksınız ve test için% 30 ayıracaksınız.

```
splits = data.randomSplit([0.7, 0.3])
train = splits[0]
test = splits[1]
train_rows = train.count()
test_rows = test.count()
print("Training Rows:", train_rows, " Testing Rows:", test_row)
```

```
Training Rows: 1892088 Testing Rows: 810130
```

Eğitim Verilerini Hazırlama

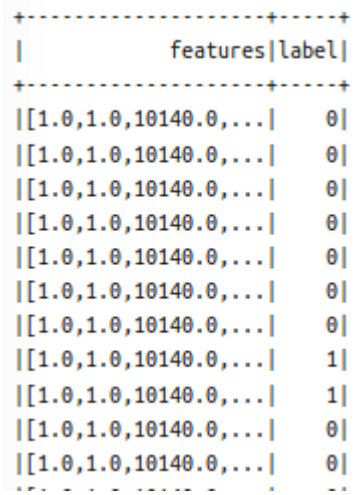
Sınıflandırma modelini eğitmek için sayısal özelliklerin bir vektörünü ve bir etiket sütunu içeren

bir eğitim veri setine ihtiyacınız vardır. Bu alıştırmada, özellik sütunlarını bir vektöre dönüştürmek için **VectorAssembler**

sınıfını kullanacaksınız ve sonra **Late** sütununu **etiket(label)** olarak yeniden adlandıracağız.

VectorAssembler, belirli bir sütun listesini tek bir vektör sütununa birleştiren bir dönüştürücüdür.

```
assembler = VectorAssembler(inputCols = ["DayofMonth", "DayOfWeek",
"OriginAirportID", "DestAirportID", "DepDelay"], outputCol="features")
training = assembler.transform(train).select(col("features"), col("Late").alias("label"))
training.show()
```



```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|[1.0,1.0,10140.0,...|    0|
|[1.0,1.0,10140.0,...|    0|
|[1.0,1.0,10140.0,...|    0|
|[1.0,1.0,10140.0,...|    0|
|[1.0,1.0,10140.0,...|    0|
|[1.0,1.0,10140.0,...|    0|
|[1.0,1.0,10140.0,...|    0|
|[1.0,1.0,10140.0,...|    1|
|[1.0,1.0,10140.0,...|    1|
|[1.0,1.0,10140.0,...|    0|
|[1.0,1.0,10140.0,...|    0|
```

Sınıflandırma Modeli Eğitim

Sonra, eğitim verilerini kullanarak bir sınıflandırma modeli eğitmeniz gerekir. Bunu yapmak için, kullanmak istediğiniz sınıflandırma algoritmasının bir örneğini yaratın ve eğitim Metodu çerçevesine dayalı bir modeli eğitmek için **fit()** yöntemini kullanın.

Bu alıştırmada, **Lojistik Regresyon** sınıflandırma algoritmasını kullanacaksınız - aynı tekniği spark.ml API'sında desteklenen sınıflandırma algoritmalarından herhangi biri için kullanabilirsiniz.

```
lr =
LogisticRegression(labelCol="label",featuresCol="features",maxIter=10,regParam=0.3
)
model = lr.fit(training)
print("Model trained!")
```

```
Model trained!
```

Test Verilerini Hazırlama

Artık eğitimli bir modeliniz var, daha önce ayırdığınız test verileri kullanılarak test edebilirsiniz. Birincisi, test verilerini, eğitim sütunundaki verilerle aynı şekilde, özellik sütunlarını bir vektöre dönüştürerek hazırlamanız gerekir.

Bu sefer, **Late** sütununu **trueLabel** olarak yeniden adlandırırsınız.

```
testing = assembler.transform(test).select(col("features"),
col("Late").alias("trueLabel"))
testing.show()
```

```
+--------------------+---------+
|            features|trueLabel|
+--------------------+---------+
|[1.0,1.0,10140.0,...|        0|
|[1.0,1.0,10140.0,...|        1|
|[1.0,1.0,10140.0,...|        0|
|[1.0,1.0,10140.0,...|        0|
|[1.0,1.0,10140.0,...|        1|
|[1.0,1.0,10140.0,...|        0|
|[1.0,1.0,10140.0,...|        1|
```

Modeli Test Edin

Şimdi bazı tahminler üretmek için modelin **transform** yöntemini kullanmaya hazırsınız. Bu yaklaşımı, etiketin bilinmediği uçuşlar için gecikme durumunu tahmin etmek için kullanabilirsiniz; ancak bu durumda bilinen bir gerçek etiket değerini içeren test verilerini kullanıyorsunuz, böylece tahmini durumu gerçek durumla karşılaştırabilirsiniz.

```
prediction = model.transform(testing)
predicted = prediction.select("features", "prediction", "probability", "trueLabel")
predicted.show(100, truncate=False)
```

```
+-------------------------------+----------+----------------------------------------+---------+
|features                       |prediction|probability                             |trueLabel|
+-------------------------------+----------+----------------------------------------+---------+
|[1.0,1.0,10140.0,10821.0,8.0]  |0.0       |[0.8064606914617916,0.19353930853820847]|0        |
|[1.0,1.0,10140.0,11259.0,-1.0] |0.0       |[0.8255955932220848,0.17440440677791524]|0        |
|[1.0,1.0,10140.0,11259.0,0.0]  |0.0       |[0.8235692331252398,0.1764307668747601]|0        |
|[1.0,1.0,10140.0,11292.0,-4.0] |0.0       |[0.8315800091597046,0.16841999084029532]|0        |
|[1.0,1.0,10140.0,11292.0,0.0]  |0.0       |[0.8235853354827378,0.17641466451726226]|0        |
|[1.0,1.0,10140.0,11298.0,-10.0]|0.0       |[0.8430292380549985,0.15697076194500142]|0        |
|[1.0,1.0,10140.0,11298.0,0.0]  |0.0       |[0.8235882630600262,0.17641173693997386]|0        |
|[1.0,1.0,10140.0,12191.0,-3.0] |0.0       |[0.8300351784480804,0.1699648215519197] |0        |
|[1.0,1.0,10140.0,12191.0,16.0] |0.0       |[0.7891320783356854,0.21086792166431453]|0        |
```

Sonuça bakıldığında, **prediction** sütununda etiket için öngörülen değer bulunur ve **trueLabel** sütunu, test verisinden bilinen gerçek değeri içerir. Doğru ve yanlış tahminlerin bir karışımı varmış gibi görünüyor.

Confusion Matrix Metrics hesapla

Sınıflandırıcılar tipik olarak sayısını belirten **confusion matrisi** oluşturarak değerlendirilir:

- Doğru Pozitifler(True Positives)

- Gerçek Negatifler(True Negatives)

- Yanlış Pozitifler(False Positive)

- Yanlış Negatifler(False Negatives)

Bu temel önlemlerden, **hassasiyet(precision)** ve **geri çağırma(recall)** gibi diğer değerlendirme metrikleri hesaplanabilir.

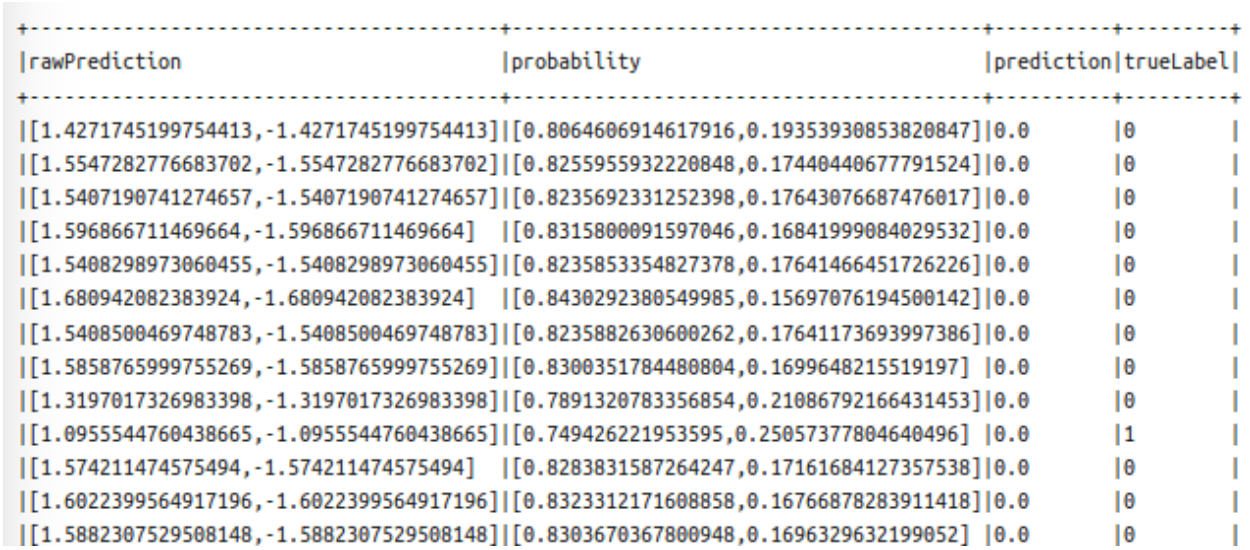
```
tp = float(predicted.filter("prediction == 1.0 AND truelabel == 1").count())
fp = float(predicted.filter("prediction == 1.0 AND truelabel == 0").count())
tn = float(predicted.filter("prediction == 0.0 AND truelabel == 0").count())
fn = float(predicted.filter("prediction == 0.0 AND truelabel == 1").count())
metrics = spark.createDataFrame([
 ("TP", tp),
 ("FP", fp),
 ("TN", tn),
 ("FN", fn),
 ("Precision", tp / (tp + fp)),
 ("Recall", tp / (tp + fn))],["metric", "value"])
metrics.show()
```

```
+---------+-------------------+
|   metric|              value|
+---------+-------------------+
|       TP|            19307.0|
|       FP|               70.0|
|       TN|           649825.0|
|       FN|           141681.0|
|Precision| 0.9963874696805491|
|   Recall|0.11992819340571968|
+---------+-------------------+
```

Tahmin ve Olasılığı Görüntüleme

Tahmin, lojistik fonksiyonda etiketli bir noktayı tanımlayan ham tahmin puanı(raw prediciton) temel alınarak yapılır. Bu ham tahmin, daha sonra, olası her etiket değeri için güveni(confidence) gösteren bir olasılık vektörüne (bu durumda, 0 ve 1) dayanan tahmin edilen bir etiket olan 0 veya 1'e dönüştürülür. En yüksek güven değeri olan tahmin tahmini olarak seçilir.

```
prediction.select("rawPrediction", "probability", "prediction", "trueLabel").show(100,
truncate=False)
```



```
+----------------------------------------+----------------------------------------+----------+---------+
|rawPrediction                           |probability                             |prediction|trueLabel|
+----------------------------------------+----------------------------------------+----------+---------+
|[1.4271745199754413,-1.4271745199754413]|[0.8064606914617916,0.19353930853820847]|0.0       |0        |
|[1.5547282776683702,-1.5547282776683702]|[0.8255955932220848,0.17440440677791524]|0.0       |0        |
|[1.5407190741274657,-1.5407190741274657]|[0.8235692331252398,0.17643076687476017]|0.0       |0        |
|[1.596866711469664,-1.596866711469664]  |[0.8315800091597046,0.16841999084029532]|0.0       |0        |
|[1.5408298973060455,-1.5408298973060455]|[0.8235853354827378,0.17641466451726226]|0.0       |0        |
|[1.680942082383924,-1.680942082383924]  |[0.8430292380549985,0.15697076194500142]|0.0       |0        |
|[1.5408500469748783,-1.5408500469748783]|[0.8235882630600262,0.17641173693997386]|0.0       |0        |
|[1.5858765999755269,-1.5858765999755269]|[0.8300351784480804,0.1699648215519197] |0.0       |0        |
|[1.3197017326983398,-1.3197017326983398]|[0.7891320783356854,0.21086792166431453]|0.0       |0        |
|[1.0955544760438665,-1.0955544760438665]|[0.749426221953595,0.25057377804640496] |0.0       |1        |
|[1.574211474575494,-1.574211474575494]  |[0.8283831587264247,0.17161684127357538]|0.0       |0        |
|[1.6022399564917196,-1.6022399564917196]|[0.8323312171608858,0.16766878283911418]|0.0       |0        |
|[1.5882307529508148,-1.5882307529508148]|[0.8303670367800948,0.1696329632199052] |0.0       |0        |
```

Sonuçların, 0 olasılığının (olasılık vektöründeki ilk değer) yalnızca 1 olasılığı (olasılık vektöründeki ikinci değer) biraz daha yüksek olduğu satırları içerdiğini unutmayın. Varsayılan ayırma eşiği (bir olasılığın 1 veya 0 olarak tahmin edilip edilmeyeceğini belirleyen sınır), 0,5 olarak ayarlanır; bu nedenle eşiğe ne kadar yakın olursa olsun, olasılıkla en yüksek ihtimalle tahmini kullanılır.

2.Lab: Pipeline Oluşturma

Lab Adımları:

Bu alıştırmada, özellikleri hazırlamak ve bir sınıflandırma modeli oluşturmak için **transformer** ve **estimator** çok aşamalarını içeren bir boru hattı uygulayacaksınız. Elde edilen eğitilmiş * PipelineModel * daha sonra bir uçuşun geç olup olmayacağını tahmin etmek için bir transformatör olarak kullanılabilir.

Spark SQL ve Spark ML Kütüphanelerini İçe Aktar

İlk önce ihtiyacınız olan kitaplıkları alın:

```
from pyspark.sql.types import *
from pyspark.sql.functions import *

from pyspark.ml import Pipeline
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.feature import VectorAssembler, StringIndexer, VectorIndexer,
MinMaxScaler
```

Veriyi Yükle

Bu alıştırma verileri, uçuşların ayrıntılarını içeren bir CSV dosyası olarak sağlanmaktadır. Veriler, her uçuş için belirli özellikleri (veya **features**) ve uçuşun kaç dakika geç geldiğini gösteren bir sütun içerir.

Bu verileri bir DataFrame'e yükleyip görüntüleyeceksiniz.

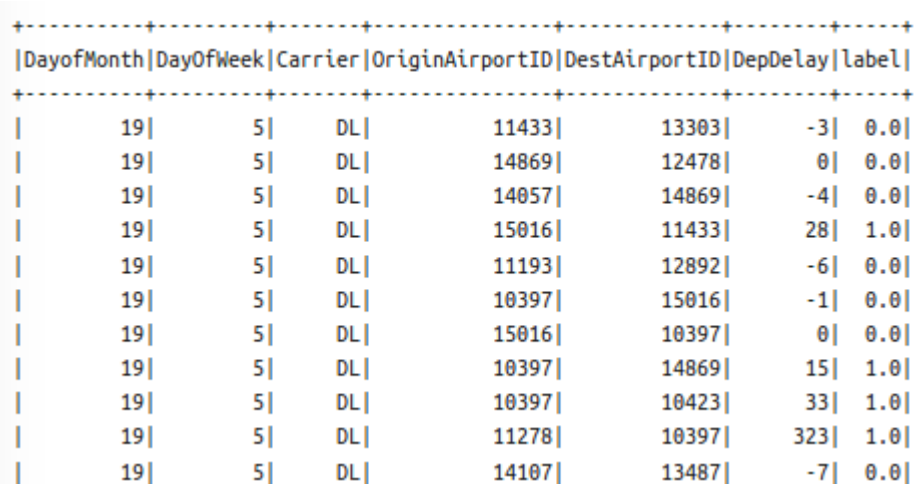
```
csv = spark.read.csv('/kaggle/input/flights-and-airports-data/flights.csv',
inferSchema=True, header=True)
csv.show()
```

```
+----------+---------+-------+--------------+-------------+--------+--------+
|DayofMonth|DayOfWeek|Carrier|OriginAirportID|DestAirportID|DepDelay|ArrDelay|
+----------+---------+-------+--------------+-------------+--------+--------+
|        19|        5|     DL|         11433|        13303|      -3|       1|
|        19|        5|     DL|         14869|        12478|       0|      -8|
|        19|        5|     DL|         14057|        14869|      -4|     -15|
|        19|        5|     DL|         15016|        11433|      28|      24|
|        19|        5|     DL|         11193|        12892|      -6|     -11|
|        19|        5|     DL|         10397|        15016|      -1|     -19|
|        19|        5|     DL|         15016|        10397|       0|      -1|
|        19|        5|     DL|         10397|        14869|      15|      24|
|        19|        5|     DL|         10397|        10423|      33|      34|
```

Verileri Hazırla

Çoğu modelleme, kapsamlı bir araştırma ve verilerin hazırlanmasıyla başlar. Bu örnekte, veriler sizin için temizlendi. **features** olarak kullanılacak sütunların bir alt kümesini seçeceksiniz ve planlanan varış saatinden sonra 15 dakika veya daha fazla sürede ulaşan uçuşlar için **1** değeri ile **label** adında bir Boolen **label** alanı oluşturacaksınız, veya uçuş erken veya zamanında ise **0**.

```
data = csv.select("DayofMonth", "DayOfWeek", "Carrier", "OriginAirportID",
"DestAirportID", "DepDelay", ((col("ArrDelay") > 15).cast("Double").alias("label")))
data.show()
```



DayofMonth	DayOfWeek	Carrier	OriginAirportID	DestAirportID	DepDelay	label
19	5	DL	11433	13303	-3	0.0
19	5	DL	14869	12478	0	0.0
19	5	DL	14057	14869	-4	0.0
19	5	DL	15016	11433	28	1.0
19	5	DL	11193	12892	-6	0.0
19	5	DL	10397	15016	-1	0.0
19	5	DL	15016	10397	0	0.0
19	5	DL	10397	14869	15	1.0
19	5	DL	10397	10423	33	1.0
19	5	DL	11278	10397	323	1.0
19	5	DL	14107	13487	-7	0.0

Verileri Böl

Kaynak verileri ayırmak için denetlenen makine öğrenme modelleri oluştururken bazılarını modeli eğitmek ve bazılarını eğitilmiş modeli test etmek için ayırmak yaygın bir uygulamadır. Bu alıştırmada, eğitim için verilerin% 70'ini kullanacaksınız ve test için% 30 ayıracaksınız. Test verisinde, etiket sütununun adı trueLabel olarak yeniden adlandırılır, böylece tahmini etiketleri bilinen gerçek değerlerle karşılaştırmak için daha sonra kullanabilirsiniz.

```
splits = data.randomSplit([0.7, 0.3])
train = splits[0]
test = splits[1].withColumnRenamed("label", "trueLabel")
train_rows = train.count()
test_rows = test.count()
print("Training Rows:", train_rows, " Testing Rows:", test_rows)
```

```
Training Rows: 1891633 Testing Rows: 810585
```

Pipeline Tanımla

Tahmini bir model genellikle çok aşamalı bir özellik hazırlama gerektirir. Örneğin, devamlı özellikler (hesaplanabilir bir sayısal değere sahip) ve kategorik özellikler (ayrık kategorilerin sayısal temsilleri olan) arasındaki farkları belirlemek için bazı algoritmalar kullanıldığında yaygın bir durumdur. Ortak bir ölçeği kullanmak için sürekli sayısal özellikleri normalleştirmek için de geçerlidir (örneğin, tüm sayıları 0 ile 1 arasında orantılı ondalık değere ölçekleyerek).

Bir pipeline, genellikle bir DataFrame hazırlayan bir dizi **transformer** ve **estimator** aşamasından oluşur.

Modelleme ve ardından tahmini bir modeli eğitme. Bu durumda yedi aşamalı bir pipeline oluşturacaksınız:

- String değerlerini kategorik özellikler için stringlere dönüştüren **StringIndexer** estimator

- Kategorik özellikleri tek bir vektöre birleştiren bir **VectorAssembler**

- Bir kategorik özellik vektörü için dizinler oluşturan **VectorIndexer**

- Bir **VectorAssembler**, sürekli sayısal özellikler içeren bir vektör oluşturur

- Sürekli sayısal özellikleri normalleştiren **MinMaxScaler**

- Kategorik ve sürekli özellikler içeren bir vektör oluşturan bir **VectorAssembler**

- Sınıflandırma modelini eğiten bir **DecisionTreeClassifier**.

```
strIdx = StringIndexer(inputCol = "Carrier", outputCol = "CarrierIdx")
catVect = VectorAssembler(inputCols = ["CarrierIdx", "DayofMonth", "DayOfWeek",
"OriginAirportID", "DestAirportID"], outputCol="catFeatures")
catIdx = VectorIndexer(inputCol = catVect.getOutputCol(), outputCol =
"idxCatFeatures")
numVect = VectorAssembler(inputCols = ["DepDelay"], outputCol="numFeatures")
minMax = MinMaxScaler(inputCol = numVect.getOutputCol(),
outputCol="normFeatures")
featVect = VectorAssembler(inputCols=["idxCatFeatures", "normFeatures"],
outputCol="features")
dt = DecisionTreeClassifier(labelCol="label", featuresCol="features")
pipeline = Pipeline(stages=[strIdx, catVect, catIdx, numVect, minMax, featVect, dt])
```

Pipeline'ı Estimator olarak çalıştırın

Boru hattının kendisi bir tahmincidir ve bu nedenle belirli bir DataFrame'de pipelineı çalıştırmak için arayabileceğiniz **fit()** yöntemi vardır. Bu durumda, bir modeli eğitmek için pipeline eğitim verileri üzerinde çalıştırırsınız.

```
piplineModel = pipeline.fit(train)
print("Pipeline complete!")
```

Pipeline Modelini Test Edin

Pipeline tarafından üretilen model, boru hattındaki tüm aşamaları belirli bir DataFrame'e uygulayacak ve tahmin modelleri oluşturmak için eğitilmiş modeli uygulayacak bir transformatördür. Bu durumda, etiket tahminlerini üretmek için pipelineı kullanarak test DataFrame'ini dönüştüreceksiniz.

```
prediction = piplineModel.transform(test)
predicted = prediction.select("features", "prediction", "trueLabel")
predicted.show(100, truncate=False)
```

```
+-----------------------------------------------------+----------+---------+
|features                                             |prediction|trueLabel|
+-----------------------------------------------------+----------+---------+
|[10.0,1.0,0.0,10397.0,12264.0,0.029889879391714735]|0.0       |0.0      |
|[10.0,1.0,0.0,10397.0,13851.0,0.030414263240692185]|0.0       |0.0      |
|[10.0,1.0,0.0,10423.0,13244.0,0.02831672784478238] |0.0       |0.0      |
|[10.0,1.0,0.0,10423.0,13487.0,0.026219192448872573]|0.0       |0.0      |
|[10.0,1.0,0.0,10423.0,14869.0,0.04929208180388044] |1.0       |1.0      |
|[10.0,1.0,0.0,10529.0,11193.0,0.028841111693759833]|0.0       |0.0      |
|[10.0,1.0,0.0,10529.0,11193.0,0.03618248557944415] |0.0       |0.0      |
|[10.0,1.0,0.0,10693.0,13487.0,0.029365495542737284]|0.0       |0.0      |
```

Ortaya çıkan DataFrame, pipline'deki tüm dönüşümleri test verilerine uygulayarak üretilir. Tahmin sütununda, etiket için öngörülen değer bulunur ve **trueLabel** sütunu, test verisinden bilinen gerçek değeri içerir.

3.Lab: Regresyon

Lab Adımları:

Regresyon Modeli

Bu alıştırmada, bir uçuşun ne kadar geç veya erken geleceğini tahmin etmek için bir uçuş özelliklerini kullanan bir regresyon modeli uygulayacaksınız.

Spark SQL and Spark ML Kütüphaleri

Öncelikle, kullanacağımız kütüphaneleri programa dahil edeceğiz;

```
from pyspark.sql.types import *
from pyspark.sql.functions import *

from pyspark.ml.regression import LinearRegression
from pyspark.ml.feature import VectorAssembler
```

Verinin Yüklenmesi

Bu alıştırma verileri, uçuşların ayrıntılarını içeren bir CSV dosyası olarak sağlanmaktadır. Veriler, her uçuşun spesifik **özellikler**ini (**feature**) ve uçuşun kaç veya son kaç dakika geç geldiğini gösteren bir **etiket** (**label**) sütununu içerir.

Bu verileri bir DataFrame'e yükleyip verileri listeleyeceksiniz.

```
csv = spark.read.csv('/data/flights.csv', inferSchema=True, header=True)
csv.show()
```

```
+----------+---------+-------+---------------+-------------+--------+--------+
|DayofMonth|DayOfWeek|Carrier|OriginAirportID|DestAirportID|DepDelay|ArrDelay|
+----------+---------+-------+---------------+-------------+--------+--------+
|        19|        5|     DL|          11433|        13303|      -3|       1|
|        19|        5|     DL|          14869|        12478|       0|      -8|
|        19|        5|     DL|          14057|        14869|      -4|     -15|
|        19|        5|     DL|          15016|        11433|      28|      24|
|        19|        5|     DL|          11193|        12892|      -6|     -11|
|        19|        5|     DL|          10397|        15016|      -1|     -19|
|        19|        5|     DL|          15016|        10397|       0|      -1|
|        19|        5|     DL|          10397|        14869|      15|      24|
|        19|        5|     DL|          10397|        10423|      33|      34|
|        19|        5|     DL|          11278|        10397|     323|     322|
```

```
csv.describe().show()
```

```
+-------+------------------+------------------+-------+-----------------+-----------------+-----------------+-----------------+
|summary|        DayofMonth|         DayOfWeek|Carrier|   OriginAirportID|    DestAirportID|         DepDelay|         ArrDelay|
+-------+------------------+------------------+-------+-----------------+-----------------+-----------------+-----------------+
|  count|           2702218|           2702218|2702218|          2702218|          2702218|          2702218|          2702218|
|   mean|15.797897875004903| 3.899480352806472|   null|12742.597593162358|12743.000197985506|10.510732294729737|6.6550108096386005|
| stddev| 8.798835069164106|1.9859246033675775|   null|1501.8408475102672|1501.8014309297678|36.029756084661265| 38.54758423679121|
|    min|                 1|                 1|     9E|            10140|            10140|              -63|              -94|
|    max|                31|                 7|     YV|            15376|            15376|             1863|             1845|
+-------+------------------+------------------+-------+-----------------+-----------------+-----------------+-----------------+
```

Verilerin Hazırlanması

Çoğu modelleme, kapsamlı bir araştırma ve verilerin hazırlanmasıyla başlar. Bu örnekte, model olarak tahmin edecek olan **ArrDelay** sütununun yanı sıra özellikler olarak tahmin işleminde kullanılacak bir sütun alt grubu seçeceksiniz.

```
data = csv.select("DayofMonth", "DayOfWeek", "OriginAirportID", "DestAirportID",
"DepDelay", "ArrDelay")
data.show()
```

```
+----------+---------+---------------+-------------+--------+--------+
|DayofMonth|DayOfWeek|OriginAirportID|DestAirportID|DepDelay|ArrDelay|
+----------+---------+---------------+-------------+--------+--------+
|        19|        5|          11433|        13303|      -3|       1|
|        19|        5|          14869|        12478|       0|      -8|
|        19|        5|          14057|        14869|      -4|     -15|
|        19|        5|          15016|        11433|      28|      24|
|        19|        5|          11193|        12892|      -6|     -11|
|        19|        5|          10397|        15016|      -1|     -19|
|        19|        5|          15016|        10397|       0|      -1|
|        19|        5|          10397|        14869|      15|      24|
|        19|        5|          10397|        10423|      33|      34|
|        19|        5|          11278|        10397|     323|     322|
|        19|        5|          14107|        13487|      -7|     -13|
```

Verilerin Bölünmesi

Kaynak verileri ayırmak için denetlenen makine öğrenme modelleri oluştururken bazılarını modeli eğitmek ve bazılarını eğitilmiş modeli test etmek için ayırmak yaygın bir uygulamadır. Bu alıştırmada, eğitim için verilerin %70'ini ayıracaksınız ve test için kalan %30'unu kullanacaksınız

```
splits = data.randomSplit([0.7, 0.3])
train = splits[0]
test = splits[1]
train_rows = train.count()
```

```
test_rows = test.count()
print("Training Rows:", train_rows, " Testing Rows:", test_rows)
```

Eğitim Verilerinin Hazırlanması

Regresyon modelini eğitmek için sayısal özelliklerin bir vektörünü ve bir etiket sütununu içeren bir eğitim veri setine ihtiyacınız vardır. Bu alıştırmada, özellik sütunlarını bir vektöre dönüştürmek için **VectorAssembler** sınıfını kullanacaksınız ve ardından **ArrDelay** sütununu **label** olarak yeniden adlandıracağız.

```
assembler = VectorAssembler(inputCols = ["DayofMonth", "DayOfWeek",
"OriginAirportID", "DestAirportID", "DepDelay"], outputCol="features")
training = assembler.transform(train).select(col("features"),
(col("ArrDelay").cast("Int").alias("label")))
training.show()
```

```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|[1.0,1.0,10140.0,...|  -17|
|[1.0,1.0,10140.0,...|   -9|
|[1.0,1.0,10140.0,...|  -11|
|[1.0,1.0,10140.0,...|  -11|
|[1.0,1.0,10140.0,...|  -12|
|[1.0,1.0,10140.0,...|   19|
|[1.0,1.0,10140.0,...|   23|
|[1.0,1.0,10140.0,...|   -8|
|[1.0,1.0,10140.0,...|   -5|
|[1.0,1.0,10140.0,...|   -6|
|[1.0,1.0,10140.0,...|   -5|
|[1.0,1.0,10140.0,...|   -1|
|[1.0,1.0,10140.0,...|    6|
```

Regresyon Modeli Oluşturma

Sonra, eğitim verilerini kullanarak bir regresyon modeli eğitmeniz gerekir. Bunu yapmak için, kullanmak istediğiniz regresyon algoritmasının bir örneğini yaratın ve eğitim için kullanılan DataFrame'e dayalı bir modeli eğitmek için **fit** yöntemini kullanın. Bu alıştırmada, **Lineer Regresyon** , (**Linear Regression**) algoritması kullanacaksınız - aynı tekniği spark.ml API'sında desteklenen herhangi bir regresyon algoritması için kullanabilirsiniz.

```
lr = LinearRegression(labelCol="label",featuresCol="features", maxIter=10,
regParam=0.3)
model = lr.fit(training)
```

```
print("Model trained!")
```

Test Verilerini Hazırlayın

Artık eğitimli bir modeliniz var, daha önce ayırdığınız test verileri kullanılarak test edebilirsiniz. Birincisi, test verilerini, eğitim sütunundaki verilerle aynı şekilde, özellik sütunlarını bir vektöre dönüştürerek hazırlamanız gerekir. Bu sefer **ArrDelay** sütununu **trueLabel** olarak yeniden adlandıracaksınız.

```
Model trained!
```

```
testing = assembler.transform(test).select(col("features"),
(col("ArrDelay")).cast("Int").alias("trueLabel"))
testing.show()
```

```
+--------------------+---------+
|            features|trueLabel|
+--------------------+---------+
|[1.0,1.0,10140.0,...|      -11|
|[1.0,1.0,10140.0,...|      -14|
|[1.0,1.0,10140.0,...|      -10|
|[1.0,1.0,10140.0,...|       41|
|[1.0,1.0,10140.0,...|      -19|
|[1.0,1.0,10140.0,...|      -13|
|[1.0,1.0,10140.0,...|       -1|
|[1.0,1.0,10140.0,...|       18|
|[1.0,1.0,10140.0,...|       -8|
```

Modelin Test Edilmesi

Şimdi bazı öngörüler üretmek için modelin ** transform ** yöntemini kullanmaya hazırsınız. Bu yaklaşımı, etiketin bilinmediği uçuşlar için varış gecikmesini öngörmek için kullanabilirsiniz; ancak bu durumda bilinen bir gerçek etiket değerini içeren test verilerini kullanıyorsunuz, böylece tahmini gecikme sayısını geç veya erken saatte gerçek varış gecikmesi ile karşılaştırabilirsiniz.

```
prediction = model.transform(testing)
predicted = prediction.select("features", "prediction", "trueLabel")
predicted = predicted.withColumn("id",monotonically_increasing_id())
predicted.show()
```

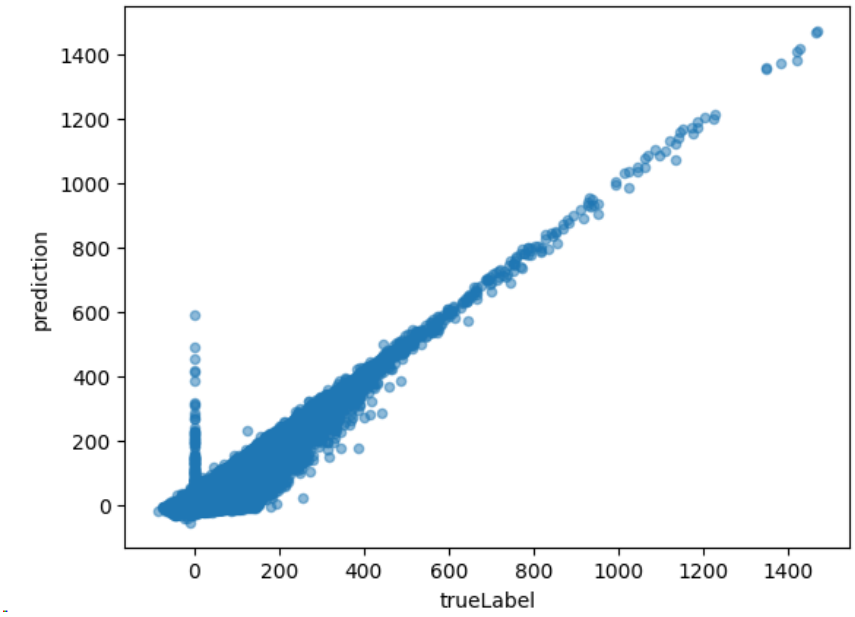
```
+--------------------+-------------------+---------+---+
|            features|         prediction|trueLabel| id|
+--------------------+-------------------+---------+---+
|[1.0,1.0,10140.0,...| -7.573506668279767|      -11|  0|
|[1.0,1.0,10140.0,...|  -5.77883441327332|      -14|  1|
|[1.0,1.0,10140.0,...| -3.784232335621922|      -10|  2|
|[1.0,1.0,10140.0,...|  31.12130402327754|       41|  3|
|[1.0,1.0,10140.0,...|-13.766288272699274|      -19|  4|
|[1.0,1.0,10140.0,...|-13.766288272699274|      -13|  5|
|[1.0,1.0,10140.0,...| -7.782482039745081|       -1|  6|
|[1.0,1.0,10140.0,...|  27.91323548370869|       18|  7|
|[1.0,1.0,10140.0,...|  -9.00429823903518|       -8|  8|
```

Sonuçların Değerlendirilmesi

Sonuçlara bakıldığında, **prediction** sütununda etiket için öngörülen değer bulunur ve **trueLabel** sütunu, test verisinden bilinen gerçek değeri içerir. Tahminler ile gerçek değerler arasında bazı farklılıklar varolduğu görünmektedir.

```
df = predicted.toPandas()

df.plot.scatter(x='trueLabel', y='prediction', alpha = 0.5)
```



Ortalama Kare Hatanın Kökünün (RMSE) Hesaplanması

Tahmini ve gerçek değerler arasındaki varyansı ölçmek için kullanılan birkaç metrik vardır. Bunlardan kök ortalama karesel hata (RMSE), tahmin edilen ve gerçek değerlerle aynı birimlerde ölçülen yaygın olarak kullanılan bir değerdir - bu durumda, RMSE, tahmin edilen ve gerçek uçuş gecikme değerleri arasındaki ortalama dakika sayısını gösterir. RMSE'yi almak için **RegressionEvaluator** sınıfını kullanabilirsini

```
from pyspark.ml.evaluation import RegressionEvaluator

evaluator = RegressionEvaluator(labelCol="trueLabel", predictionCol="prediction",
metricName="rmse")
rmse = evaluator.evaluate(prediction)
print("Root Mean Square Error (RMSE):", rmse)
```

```
Root Mean Square Error (RMSE): 13.2320114798
```