BIG DATA IN MACHINE LEARNING

Bài 7: PySpark ML – Linear & Logistic Regression, Pipeline

Phòng LT & Mạng

https://csu.edu.vn/lap-trinh-co-ban/Big-Data-in-Machine-Learning-198

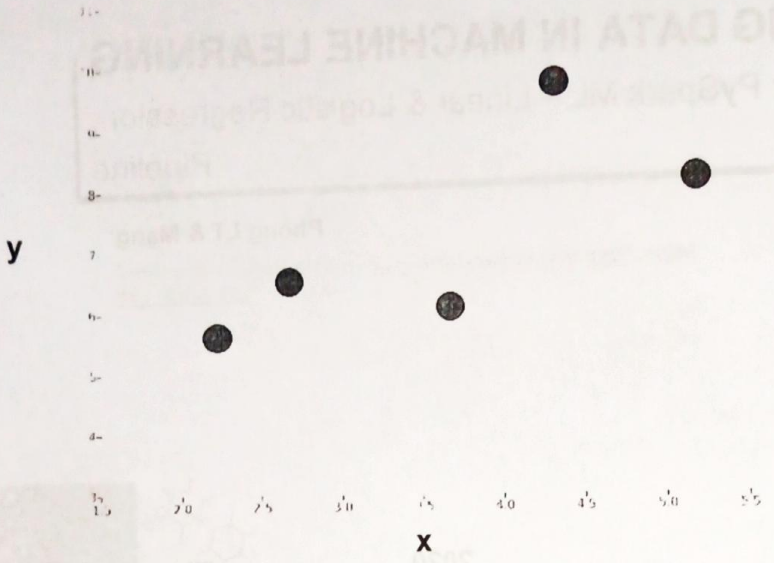
2020

Nội dung

1. **Linear Regression**
2. **Logistic Regression**
3. **Pipeline**

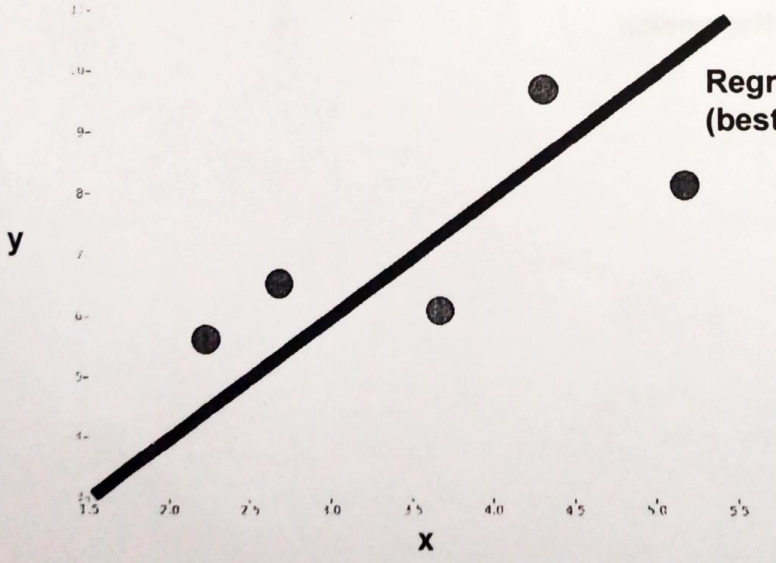
Linear Regression

- x vs. y: scatter



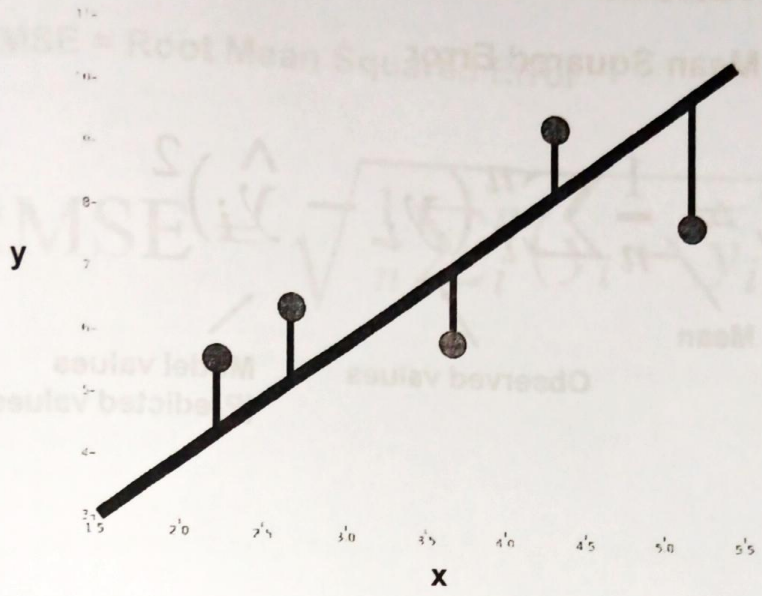
Linear Regression

- x vs. y: regression line

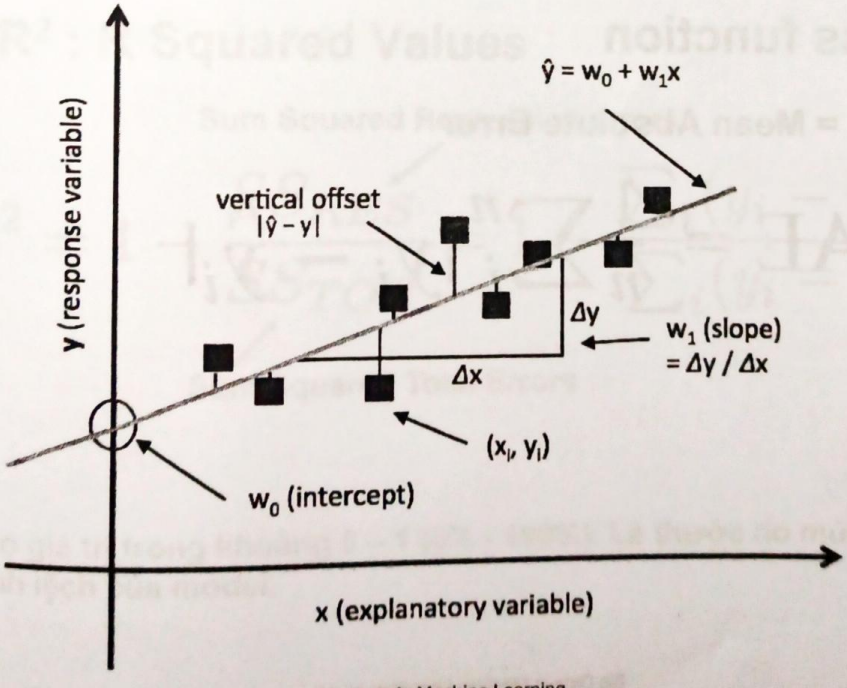


Linear Regression

- x vs. y: residual



Linear Regression



Linear Regression

❑Loss function

MSE = Mean Squared Error

$$\mathrm{MSE} = \frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2$$

Mean

Observed values

Model values
(Predicted values)

Linear Regression

❑Loss function

MAE = Mean Absolute Error

$$\mathrm{MAE} = \frac{1}{n}\sum_{i}^{n}|y_i - \hat{y}_i|$$

Linear Regression

❑Loss function

RMSE = Root Mean Squared Error

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_i^n (y_i - \hat{y}_i)^2}$$

Linear Regression

❑R^2 : R Squared Values

Sum Squared Regression errors

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

Sum Squared Total Errors

R2 có giá trị trong khoảng 0 – 1 (0% - 100%). Là thước đo mức độ chênh lệch của model.

Linear Regression

❑ Triển khai Linear Regression

- pyspark.ml.regression.LinearRegression

http://localhost:8888/notebooks/Chapter7/demo_Linear_Regression_flights_new.ipynb

Linear Regression

- Ví dụ: Xây dựng regression model để dự đoán flight duration từ distance (mile)
  - Đọc dữ liệu

```
data.show(3)
```

```
+---+---+---+-------+------+---+----+------+--------+-----+
|mon|dom|dow|carrier|flight|org|mile|depart|duration|delay|
+---+---+---+-------+------+---+----+------+--------+-----+
| 11| 20|  6|     US|    19|JFK|2153|  9.48|     351|   NA|
|  0| 22|  2|     UA|  1107|ORD| 316| 16.33|      82|   30|
|  2| 20|  4|     UA|   226|SFO| 337|  6.17|      82|   -8|
+---+---+---+-------+------+---+----+------+--------+-----+
only showing top 3 rows
```

Linear Regression

- Chuyển đổi dữ liệu

```
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler

assembler = VectorAssembler(
    inputCols=['mile'],
    outputCol="features") # inputs

data_pre = assembler.transform(data)

data_pre.show(2)
```

```
+---+---+---+-------+------+---+----+------+--------+-----+--------+
|mon|dom|dow|carrier|flight|org|mile|depart|duration|delay|features|
+---+---+---+-------+------+---+----+------+--------+-----+--------+
| 11| 20|  6|     US|    19|JFK|2153|  9.48|     351|   NA|[2153.0]|
|  0| 22|  2|     UA|  1107|ORD| 316| 16.33|      82|   30| [316.0]|
+---+---+---+-------+------+---+----+------+--------+-----+--------+
only showing top 2 rows
```

```
final_data = data_pre.select("features","duration")
```


Linear Regression

- Chia dữ liệu train-test

```
final_data = data_pre.select("features","duration")

final_data.show(5)
```

```
+--------+--------+
|features|duration|
+--------+--------+
|[2153.0]|     351|
| [316.0]|      82|
| [337.0]|      82|
|[1236.0]|     195|
| [258.0]|      65|
+--------+--------+
only showing top 5 rows
```

```
train_data,test_data = final_data.randomSplit([0.8,0.2])
```

Linear Regression

- Xây dựng model

```
from pyspark.ml.regression import LinearRegression

# Create a Linear Regression Model object
lr = LinearRegression(featuresCol='features',
                      labelCol='duration',
                      predictionCol='prediction')

# Fit the model to the data and call this model lrModel
lrModel = lr.fit(train_data,)

# Print the coefficients and intercept for linear regression
print("Coefficients: {} Intercept: {}".format(lrModel.coefficients,lrModel.intercept))
```

```
Coefficients: [0.12165952817733405] Intercept: 44.42377032545563
```

Linear Regression

- Đánh giá kết quả

```
# Check test dataset
test_model = lrModel.transform(test_data)

# Inspect results
test_model.select("prediction", "duration").show(5)
```

```
+-----------------+--------+
|       prediction|duration|
+-----------------+--------+
|52.57495871333701|      44|
|52.57495871333701|      44|
|52.57495871333701|      46|
|52.57495871333701|      46|
|52.57495871333701|      46|
+-----------------+--------+
only showing top 5 rows
```

```
print("RMSE: {}".format(test_results.rootMeanSquaredError))
print("MSE: {}".format(test_results.meanSquaredError))
print("r2: {}".format(test_results.r2))
```

```
RMSE: 16.98071818173
MSE: 288.3447899673359
r2: 0.9624270285237152
```

- Good result!!!

Linear Regression

- Đánh giá kết quả (tt)

```
from pyspark.ml.evaluation import RegressionEvaluator

RegressionEvaluator(labelCol='duration').evaluate(test_model)
```

```
16.98071818173
```

Linear Regression

- Lưu và load model

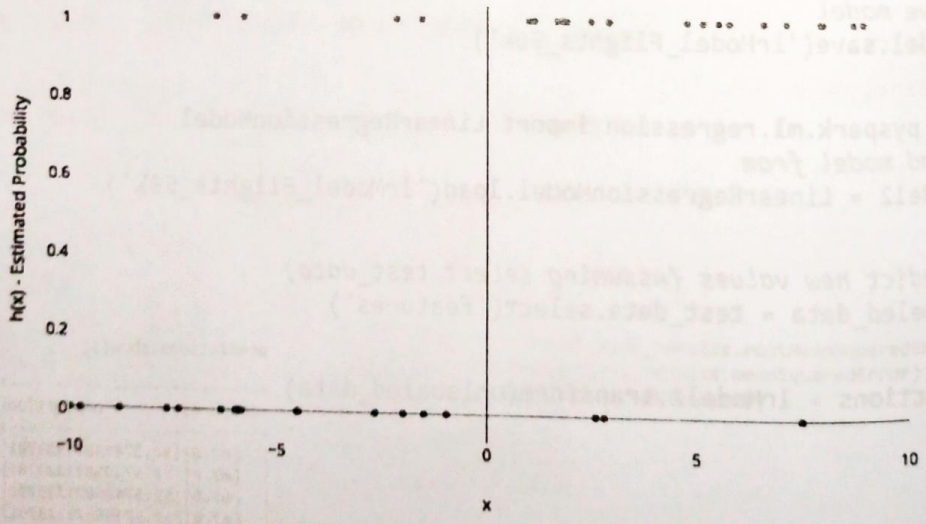
```
# Save model
lrModel.save('lrModel_Flights_50k')

from pyspark.ml.regression import LinearRegressionModel
# Load model from
lrModel2 = LinearRegressionModel.load('lrModel_Flights_50k')

# Predict new values (Assuming select test_data)
unlabeled_data = test_data.select('features')

predictions = lrModel2.transform(unlabeled_data)
```

```
predictions.show(5)
+--------+-----------------+
|features|       prediction|
+--------+-----------------+
|  [67.0]|52.57495871333701|
|  [67.0]|52.57495871333701|
|  [67.0]|52.57495871333701|
|  [67.0]|52.57495871333701|
|  [67.0]|52.57495871333701|
+--------+-----------------+
only showing top 5 rows
```

Nội dung

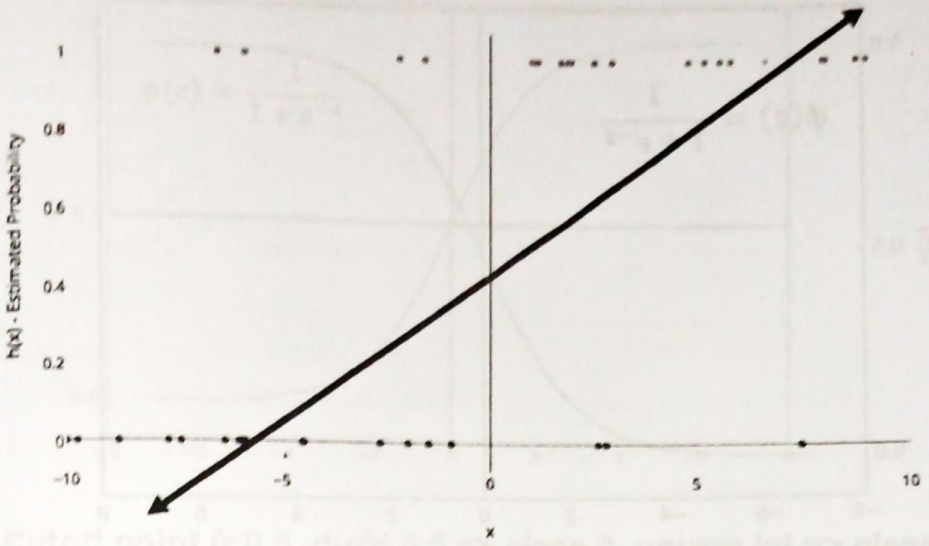
1. Linear Regression
2. Logistic Regression
3. Pipeline

Logistic Regression



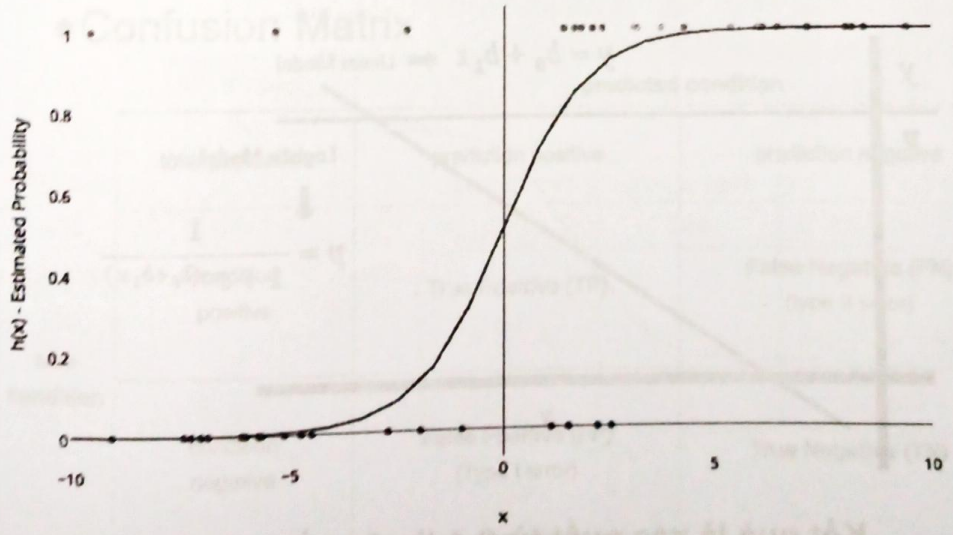
Logistic Regression

- Linear regression problem



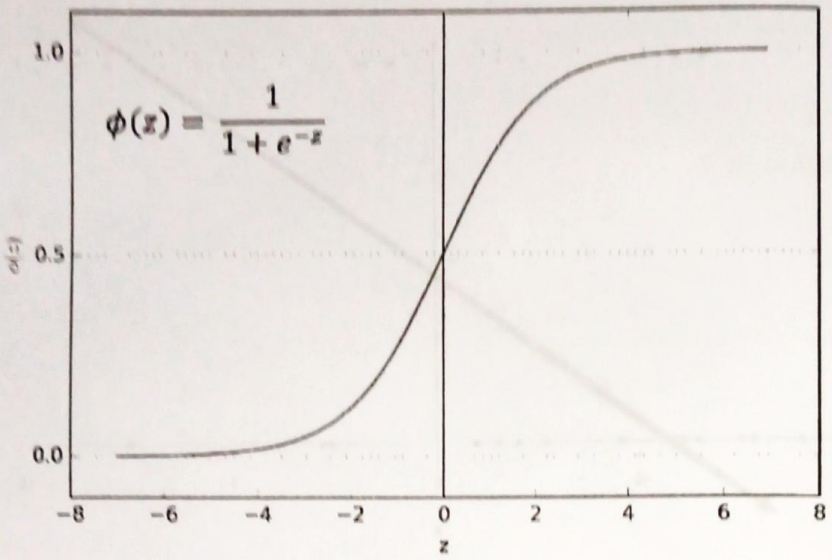
Logistic Regression

- Solution



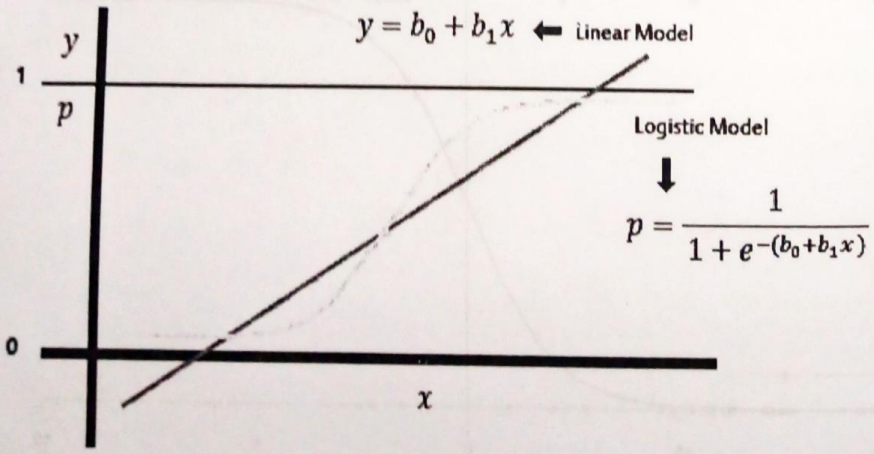
Logistic Regression

❑Sigmoid function



Logistic Regression

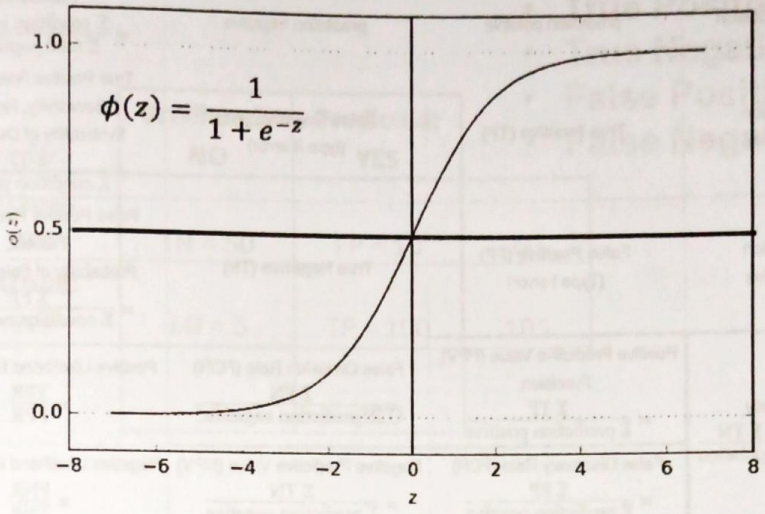
❑Sigmoid function



Kết quả là xác suất từ 0-1 thuộc về 1 class

Logistic Regression

❑ Sigmoid function



Cutoff point ở 0.5, dưới 0.5 => class 0, ngược lại => class 1

Logistic Regression

❑ Model evaluation

- Confusion Matrix

		predicted condition	
	total population	prediction positive	prediction negative
true condition	condition positive	**True Positive (TP)**	**False Negative (FN)** (type II error)
	condition negative	**False Positive (FP)** (Type I error)	**True Negative (TN)**

Logistic Regression

		predicted condition		
	total population	prediction positive	prediction negative	Prevalence $= \frac{\Sigma \text{ condition positive}}{\Sigma \text{ total population}}$
true condition	condition positive	**True Positive (TP)**	**False Negative (FN)** (type II error)	True Positive Rate (TPR), Sensitivity, Recall, Probability of Detection $= \frac{\Sigma \text{ TP}}{\Sigma \text{ condition positive}}$
	condition negative	**False Positive (FP)** (Type I error)	**True Negative (TN)**	False Positive Rate (FPR), Fall-out, Probability of False Alarm $= \frac{\Sigma \text{ FP}}{\Sigma \text{ condition negative}}$
	Accuracy $= \frac{\Sigma \text{ TP} + \Sigma \text{ TN}}{\Sigma \text{ total population}}$	Positive Predictive Value (PPV), Precision $= \frac{\Sigma \text{ TP}}{\Sigma \text{ prediction positive}}$	False Omission Rate (FOR) $= \frac{\Sigma \text{ FN}}{\Sigma \text{ prediction negative}}$	Positive Likelihood Ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$
		False Discovery Rate (FDR) $= \frac{\Sigma \text{ FP}}{\Sigma \text{ prediction positive}}$	Negative Predictive Value (NPV) $= \frac{\Sigma \text{ TN}}{\Sigma \text{ prediction negative}}$	Negative Likelihood Ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$

https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Logistic Regression

- Ví dụ

Example: Test for presence of disease

- **NO = negative test = False = 0**
- **YES = positive test = True = 1**

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

Logistic Regression

- Ví dụ

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Basic Terminology:

- **True Positives (TP)**
- **True Negatives (TN)**
- **False Positives (FP)**
- **False Negatives (FN)**

Logistic Regression

- Ví dụ

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Accuracy:

- **Overall, how often is it correct?**
- **(TP + TN) / total = 150/165 = 0.91**

Logistic Regression

- Ví dụ

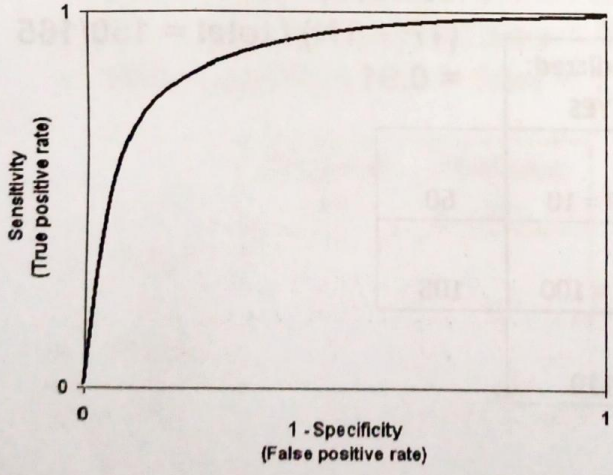
Misclassification Rate (Error Rate):

- **Overall, how often is it wrong?**
- **(FP + FN) / total = 15/165 = 0.09**

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Logistic Regression

- Receiver Operator Curve (ROC)



$$\text{TPR (sensitivity)} = \frac{TP}{TP + FN}$$

$$\text{FPR (1-specificity)} = \frac{FP}{TN + FP}$$

Logistic Regression

❑ Triển khai Logistic Regression

- pyspark.ml.classification.LogisticRegression

http://localhost:8888/notebooks/Chapter7/demo_Logistic_Regression_flights_now.ipynb

Logistic Regression

- Ví dụ: Xây dựng logistic model để dự đoán flight delay từ 'mon', 'dom', 'dow', 'carrier', 'org', 'km' (= mile * 1.60934), 'depart', 'duration'
  - Đọc dữ liệu

```
data.show(3)
```

```
+---+---+---+-------+------+---+----+------+--------+-----+
|mon|dom|dow|carrier|flight|org|mile|depart|duration|delay|
+---+---+---+-------+------+---+----+------+--------+-----+
| 11| 20|  6|     US|    19|JFK|2153|  9.48|     351|   NA|
|  0| 22|  2|     UA|  1107|ORD| 316| 16.33|      82|   30|
|  2| 20|  4|     UA|   226|SFO| 337|  6.17|      82|   -8|
+---+---+---+-------+------+---+----+------+--------+-----+
only showing top 3 rows
```

Logistic Regression

- Chuẩn hóa dữ liệu: "mile" => "km", tạo cột label từ cột "delay", chuyển "carrier" và "org" thành dữ liệu category

```
# Import the required function
from pyspark.sql.functions import round

# Convert 'mile' to 'km'
data = data.withColumn('km', round(data.mile * 1.60934, 0))

# Create 'label' column indicating whether flight delayed (1) or not (0)
data = data.withColumn('label', (data.delay >= 15).cast('integer'))

data.show(3)
```

```
+---+---+---+-------+---+----+------+--------+-----+------+-----+
|mon|dom|dow|carrier|org|mile|depart|duration|delay|    km|label|
+---+---+---+-------+---+----+------+--------+-----+------+-----+
| 11| 20|  6|     US|JFK|2153|  9:48|     351|   NA|3465.0| null|
|  0| 22|  2|     UA|ORD| 316| 16.33|      82|   30| 509.0|    1|
|  2| 20|  4|     UA|SFO| 337|  6.17|      82|   -8| 542.0|    0|
+---+---+---+-------+---+----+------+--------+-----+------+-----+
only showing top 3 rows
```


Logistic Regression

- Chuẩn hóa dữ liệu (tt)

```
from pyspark.ml.feature import StringIndexer

# Create an indexer
indexer = StringIndexer(inputCol='carrier', outputCol='carrier_idx')

# Indexer identifies categories in the data
indexer_model = indexer.fit(data)

# Indexer creates a new column with numeric index values
data_indexed = indexer_model.transform(data)

# Repeat the process for the other categorical feature
data_indexed = StringIndexer(inputCol='org', outputCol='org_idx').fit(data_indexed).transform(data_indexed)

data_indexed.show(3)
```

```
+---+---+---+-------+---+----+------+--------+-----+------+-----+-----------+-------+
|mon|dom|dow|carrier|org|mile|depart|duration|delay|    km|label|carrier_idx|org_idx|
+---+---+---+-------+---+----+------+--------+-----+------+-----+-----------+-------+
| 11| 20|  6|     US|JFK|2153|  9.48|     351|   NA|3465.0| null|        6.0|    2.0|
|  0| 22|  2|     UA|ORD| 316| 16.33|      82|   30| 509.0|    1|        0.0|    0.0|
|  2| 20|  4|     UA|SFO| 337|  6.17|      82|   -8| 542.0|    0|        0.0|    1.0|
+---+---+---+-------+---+----+------+--------+-----+------+-----+-----------+-------+
only showing top 3 rows
```

Logistic Regression

- Chuyển đổi dữ liệu

```
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler

# Create an assembler object
assembler = VectorAssembler(inputCols=[
    'mon', 'dom', 'dow', 'carrier_idx', 'org_idx', 'km', 'depart', 'duration'
], outputCol='features')

data_pre = assembler.transform(data_indexed)

data_pre.show(3, False)
```

```
+---+---+---+-------+---+----+------+--------+-----+------+-----+-----------+-------+-----------------------------------------------+
|mon|dom|dow|carrier|org|mile|depart|duration|delay|km    |label|carrier_idx|org_idx|features                                       |
+---+---+---+-------+---+----+------+--------+-----+------+-----+-----------+-------+-----------------------------------------------+
|11 |20 |6  |US     |JFK|2153|9.48  |351     |NA   |3465.0|null |6.0        |2.0    |[11.0,20.0,6.0,6.0,2.0,3465.0,9.48,351.0]      |
|0  |22 |2  |UA     |ORD|316 |16.33 |82      |30   |509.0 |1    |0.0        |0.0    |[0.0,22.0,2.0,0.0,0.0,509.0,16.33,82.0]        |
|2  |20 |4  |UA     |SFO|337 |6.17  |82      |-8   |542.0 |0    |0.0        |1.0    |[2.0,20.0,4.0,0.0,1.0,542.0,6.17,82.0]         |
+---+---+---+-------+---+----+------+--------+-----+------+-----+-----------+-------+-----------------------------------------------+
only showing top 3 rows
```


Logistic Regression

- Chia dữ liệu train-test

```
final_data = data_pre.select("features","label")

final_data.show(5, False)
```

```
+-----------------------------------------+-----+
|features                                 |label|
+-----------------------------------------+-----+
|[0.0,22.0,2.0,0.0,0.0,509.0,16.33,82.0]  |1    |
|[2.0,20.0,4.0,0.0,1.0,542.0,6.17,82.0]   |0    |
|[9.0,13.0,1.0,1.0,0.0,1989.0,10.33,195.0]|0    |
|[5.0,2.0,1.0,0.0,1.0,885.0,7.98,102.0]   |0    |
|[7.0,2.0,6.0,1.0,0.0,1180.0,10.83,135.0] |1    |
+-----------------------------------------+-----+
only showing top 5 rows
```

```
train_data,test_data = final_data.randomSplit([0.8,0.2])
```

Logistic Regression

- Xây dựng model

```
# Import the logistic regression class
from pyspark.ml.classification import LogisticRegression

# Create a Logistic Regression Model object
logistic = LogisticRegression(featuresCol='features',
                              labelCol='label',
                              predictionCol='prediction')

# Fit the model to the data and call this model logisticModel
logisticModel = logistic.fit(train_data)
```

Logistic Regression

- Đánh giá kết quả

```
# Create predictions for the testing data and show confusion matrix
test_model = logisticModel.transform(test_data)
test_model.groupBy('label', 'prediction').count().show()
```

```
+-----+----------+-----+
|label|prediction|count|
+-----+----------+-----+
|    1|       0.0| 1690|
|    0|       0.0| 2513|
|    1|       1.0| 3082|
|    0|       1.0| 2031|
+-----+----------+-----+
```

Logistic Regression

Đánh giá kết quả (tt)

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator, BinaryClassificationEvaluator

# Calculate the elements of the confusion matrix
TN = test_model.filter('prediction = 0 AND label = prediction').count()
TP = test_model.filter('prediction = 1 AND label = prediction').count()
FN = test_model.filter('prediction = 0 AND label != prediction').count()
FP = test_model.filter('prediction = 1 AND label != prediction').count()

# Calculate precision and recall
precision = TP / (TP + FP)
recall = TP / (TP + FN)
print('precision = {:.2f}\nrecall    = {:.2f}'.format(precision, recall))
```

```
precision = 0.61
recall    = 0.65
```


Logistic Regression

Đánh giá kết quả (tt)

```
# Find weighted precision
multi_evaluator = MulticlassClassificationEvaluator()
weighted_precision = multi_evaluator.evaluate(test_model,
                                              {multi_evaluator.metricName: "weightedPrecision"})
```

```
weighted_precision
```

```
0.6058708866257614
```

```
# Find AUC
binary_evaluator = BinaryClassificationEvaluator()
auc = binary_evaluator.evaluate(test_model,
                                {binary_evaluator.metricName: "areaUnderROC"})
```

```
auc
```

```
0.6448869880760812
```

Logistic Regression

- Lưu và load model

```
# Save model
logisticModel.save('logisticModel_Flights_50k')

from pyspark.ml.classification import LogisticRegressionModel
# Load model from
logisticModel2 =LogisticRegressionModel.load('logisticModel_Flights_50k')

# Predict new values (Assuming select test_data)
unlabeled_data = test_data.select('features')

predictions = logisticModel2.transform(unlabeled_data)

predictions.show(2, False)
```

```
+-----------------------------+-----------------------------------------+-----------------------------------------+------
-----+
|features                     |rawPrediction                            |probability                              |predic
tion|
+-----------------------------+-----------------------------------------+-----------------------------------------+------
-----+
|(8,[1,5,6,7],[6.0,386.0,13.17,68.0])|[-0.42703686260247886,0.42703686260247886]|[0.3948341229473289,0.6051658770526792]|1.0
|
|(8,[1,5,6,7],[6.0,538.0,20.0,84.0]) |[-1.0077194953562736,1.0077194953562736]  |[0.2674263862886751,0.7325736137113249]|1.0
|
+-----------------------------+-----------------------------------------+-----------------------------------------+------
-----+
only showing top 2 rows
```


Nội dung

1. Linear Regression
2. Logistic Regression
3. **Pipeline**

Pipeline

- Một machine learning pipeline mô tả hoặc mô hình hóa quy trình Machine Learning: code, release, thực hiện trích xuất dữ liệu, tạo mô hình huấn luyện và điều chỉnh thuật toán.

- Một ML pipeline phải là một quá trình liên tục khi một nhóm làm việc trên nền tảng ML.

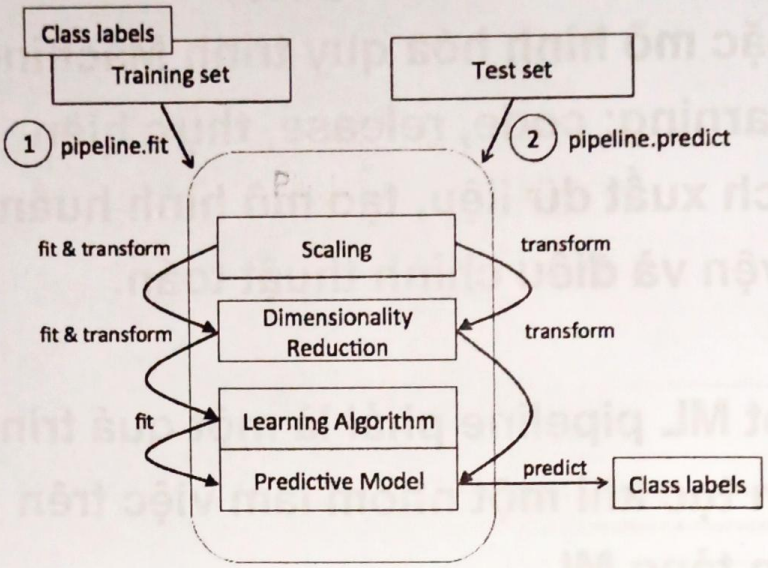
Pipeline

- Một pipline là một loạt các thao tác cần thực hiện.

- Chúng ta có thể áp dụng từng thao tác riêng lẻ hoặc có thể áp dụng pipeline

Pipeline

- Ví dụ:



Pipeline

❑ Triển khai Pipeline

- Pyspark.ml.Pipeline

http://localhost:8888/notebooks/Chapter7/demo_Pipeline_LR_flights_new.ipynb

Pipeline

- Ví dụ: Xây dựng linear model để dự đoán flight duration từ 'dow', 'org', 'km' (= mile * 1.60934),
  - Đọc dữ liệu

```
data.show(3)
```

```
+---+---+---+-------+------+---+----+------+--------+-----+
|mon|dom|dow|carrier|flight|org|mile|depart|duration|delay|
+---+---+---+-------+------+---+----+------+--------+-----+
| 11| 20|  6|     US|    19|JFK|2153|  9.48|     351|   NA|
|  0| 22|  2|     UA|  1107|ORD| 316| 16.33|      82|   30|
|  2| 20|  4|     UA|   226|SFO| 337|  6.17|      82|   -8|
+---+---+---+-------+------+---+----+------+--------+-----+
only showing top 3 rows
```


Pipeline

- Chuẩn hóa dữ liệu: “mile” => “km”

```
# Import the required function
from pyspark.sql.functions import round

# Convert 'mile' to 'km'
data = data.withColumn('km', round(data.mile * 1.60934, 0))

final_data = data
final_data.show(5)
```

```
+---+---+---+-------+---+----+------+--------+-----+------+-----+
|mon|dom|dow|carrier|org|mile|depart|duration|delay|    km|label|
+---+---+---+-------+---+----+------+--------+-----+------+-----+
|  0| 22|  2|     UA|ORD| 316| 16.33|      82|   30| 509.0|    1|
|  2| 20|  4|     UA|SFO| 337|  6.17|      82|   -8| 542.0|    0|
|  9| 13|  1|     AA|ORD|1236| 10.33|     195|   -5|1989.0|    0|
|  5|  2|  1|     UA|SFO| 550|  7.98|     102|    2| 885.0|    0|
|  7|  2|  6|     AA|ORD| 733| 10.83|     135|   54|1180.0|    1|
+---+---+---+-------+---+----+------+--------+-----+------+-----+
only showing top 5 rows
```

Pipeline

- Chia dữ liệu train-test

```
train_data,test_data = final_data.randomSplit([0.8,0.2])
```

Pipeline

- Xây dựng Pipeline model

```
# Import class for creating a pipeline
from pyspark.ml import Pipeline


# Convert categorical strings to index values
indexer = StringIndexer(inputCol='org', outputCol='org_idx')

# One-hot encode index values
onehot = OneHotEncoderEstimator(
    inputCols=['org_idx', 'dow'],
    outputCols=['org_dummy', 'dow_dummy']
)

# Assemble predictors into a single column
assembler = VectorAssembler(inputCols=['km', 'org_dummy', 'dow_dummy'], outputCol='features')

# A linear regression object
regression = LinearRegression(featuresCol='features',
                              labelCol='duration',
                              predictionCol='prediction')
```

Pipeline

- Xây dựng Pipeline model (tt)

```
# Construct a pipeline
pipeline = Pipeline(stages=[indexer, onehot, assembler, regression])

# Train the pipeline on the training data
pipeline = pipeline.fit(train_data)
```


Pipeline

- Đánh giá kết quả

```
# Make predictions on the testing data
predictions = pipeline.transform(test_data)

# Inspect results
predictions.select("prediction", "duration").show(5)
```

```
+-----------------+--------+
|       prediction|duration|
+-----------------+--------+
|363.7613046467278|     370|
|363.7613046467278|     379|
|193.1561841809605|     185|
|228.0668563212393|     250|
|72.14208567984129|      80|
+-----------------+--------+
only showing top 5 rows
```

```
from pyspark.ml.evaluation import RegressionEvaluator

RegressionEvaluator(labelCol='duration').evaluate(predictions)
```

```
11.005334277534063
```

Chapter 7: Linear Regression

Ex1: Consulting Project

Congratulations! You've been contracted by Hyundai Heavy Industries to help them build a predictive model for some ships. Hyundai Heavy Industries (http://www.hyundai.eu/en) is one of the world's largest ship manufacturing companies and builds cruise liners.

You've been flown to their headquarters in Ulsan, South Korea to help them give accurate estimates of how many crew members a ship will require.

They are currently building new ships for some customers and want you to create a model and use it to predict how many crew members the ships will need.

Here is what the data looks like so far:

```
Description: Measurements of ship size, capacity, crew, and age for 158
 cruise
ships.


Variables/Columns
Ship Name     1-20
Cruise Line   21-40
Age (as of 2013)   46-48
Tonnage (1000s of tons)   50-56
passengers (100s)   58-64
Length (100s of feet)  66-72
Cabins  (100s)   74-80
Passenger Density   82-88
Crew  (100s)   90-96
```

It is saved in a csv file for you called "cruise_ship_info.csv". Your job is to create a regression model that will help predict how many crew members will be needed for future ships. The client also mentioned that they have found that particular cruise lines will differ in acceptable crew counts, so it is most likely an important feature to include in your analysis!

Once you've created the model and tested it for a quick check on how well you can expect it to perform, make sure you take a look at why it performs so well!

Chapter 7: Linear Regression

Ex1: Consulting Project - SOLUTIONS

Congratulations! You've been contracted by Hyundai Heavy Industries to help them build a predictive model for some ships. Hyundai Heavy industries (http://www.hyundai.eu/en) is one of the world's largest ship manufacturing companies and builds cruise liners.

You've been flown to their headquarters in Ulsan, South Korea to help them give accurate estimates of how many crew members a ship will require.

They are currently building new ships for some customers and want you to create a model and use it to predict how many crew members the ships will need.

Here is what the data looks like so far:

```
Description: Measurements of ship size, capacity, crew, and age for 158
 cruise
ships.


Variables/Columns
Ship Name     1-20
Cruise Line   21-40
Age (as of 2013)   46-48
Tonnage (1000s of tons)   50-56
passengers (100s)   58-64
Length (100s of feet)  66-72
Cabins  (100s)   74-80
Passenger Density   82-88
Crew  (100s)   90-96
```

It is saved in a csv file for you called "cruise_ship_info.csv". Your job is to create a regression model that will help predict how many crew members will be needed for future ships. The client also mentioned that they have found that particular cruise lines will differ in acceptable crew counts, so it is most likely an important feature to include in your analysis!

In [1]:
```
import findspark
findspark.init()
```

In [2]:
```
from pyspark.sql import SparkSession
```

In [3]:
```
spark = SparkSession.builder.appName('cruise').getOrCreate()
```

In [4]:
```
df = spark.read.csv('cruise_ship_info.csv',inferSchema=True,header=True)
```

In [5]:
```
df.count()
```

Out[5]:
```
158
```

In [6]:
```
df.printSchema()
```

```
root
 |-- Ship_name: string (nullable = true)
 |-- Cruise_line: string (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Tonnage: double (nullable = true)
 |-- passengers: double (nullable = true)
 |-- length: double (nullable = true)
 |-- cabins: double (nullable = true)
 |-- passenger_density: double (nullable = true)
 |-- crew: double (nullable = true)
```

In [7]:
```
df.show(5)
```

```
+-----------+-----------+---+-----------------+----------+------+------+------
-----------+----+
|  Ship_name|Cruise_line|Age|          Tonnage|passengers|length|cabins|passen
ger_density|crew|
+-----------+-----------+---+-----------------+----------+------+------+------
-----------+----+
|    Journey|    Azamara|  6|30.276999999999997|      6.94|  5.94|  3.55|
42.64|3.55|
|      Quest|    Azamara|  6|30.276999999999997|      6.94|  5.94|  3.55|
42.64|3.55|
|Celebration|   Carnival| 26|           47.262|     14.86|  7.22|  7.43|
31.8| 6.7|
|   Conquest|   Carnival| 11|            110.0|     29.74|  9.53| 14.88|
36.99|19.1|
|    Destiny|   Carnival| 17|          101.353|     26.42|  8.92| 13.21|
38.36|10.0|
+-----------+-----------+---+-----------------+----------+------+------+------
-----------+----+
only showing top 5 rows
```

In [8]:
```
df.describe().show()
```

```
+-------+---------+----------+-----------------+-----------------+-----------------+-----------------+
|summary|Ship_name|Cruise_line|              Age|          Tonnage|       passengers|           length|
+-------+---------+----------+-----------------+-----------------+-----------------+-----------------+
|  count|      158|       158|              158|              158|              158|              158|
|   mean| Infinity|      null|15.689873417721518| 71.28467088607599|18.457405063291142|8.130632911392404|
| stddev|      NaN|      null| 7.615691058751413|37.229540025907866|9.677094775143416|1.793473548054825|
|    min|Adventure|   Azamara|                4|            2.329|             0.66|             2.79|
|    max|Zuiderdam|  Windstar|               48|            220.0|             54.0|            11.82|
+-------+---------+----------+-----------------+-----------------+-----------------+-----------------+

+-------+-----------------+-----------------+-----------------+
|summary|            cabins|passenger_density|             crew|
+-------+-----------------+-----------------+-----------------+
|  count|              158|              158|              158|
|   mean|15.689873417721518| 71.28467088607599|8.130632911392404|
| stddev| 7.615691058751413|37.229540025907866|1.793473548054825|
|    min|                4|            2.329|             0.66|
|    max|               48|            220.0|             54.0|
+-------+-----------------+-----------------+-----------------+
```

Dealing with the Cruise_line categorical variable

Ship Name is a useless arbitrary string, but the cruise_line itself may be useful. Let's make it into a categorical variable!

In [9]:
```
df.groupBy('Cruise_line').count().show()
```

```
+-----------------+-----+
|      Cruise_line|count|
+-----------------+-----+
|           Costa|   11|
|             P&O|    6|
|          Cunard|    3|
|Regent_Seven_Seas|    5|
|              MSC|    8|
|         Carnival|   22|
|          Crystal|    2|
|           Orient|    1|
|         Princess|   17|
|        Silversea|    4|
|         Seabourn|    3|
| Holland_American|   14|
|         Windstar|    3|
|           Disney|    2|
|        Norwegian|   13|
|          Oceania|    3|
|          Azamara|    2|
|        Celebrity|   10|
|             Star|    6|
|  Royal_Caribbean|   23|
+-----------------+-----+
```

In [10]:
```
from pyspark.ml.feature import StringIndexer
indexer = StringIndexer(inputCol="Cruise_line", outputCol="cruise_cat")
indexed = indexer.fit(df).transform(df)
indexed.head(5)
```

Out[10]:
```
[Row(Ship_name='Journey', Cruise_line='Azamara', Age=6, Tonnage=30.276999999999
997, passengers=6.94, length=5.94, cabins=3.55, passenger_density=42.64, crew=
3.55, cruise_cat=16.0),
 Row(Ship_name='Quest', Cruise_line='Azamara', Age=6, Tonnage=30.27699999999999
7, passengers=6.94, length=5.94, cabins=3.55, passenger_density=42.64, crew=3.5
5, cruise_cat=16.0),
 Row(Ship_name='Celebration', Cruise_line='Carnival', Age=26, Tonnage=47.262, p
assengers=14.86, length=7.22, cabins=7.43, passenger_density=31.8, crew=6.7, cr
uise_cat=1.0),
 Row(Ship_name='Conquest', Cruise_line='Carnival', Age=11, Tonnage=110.0, passe
ngers=29.74, length=9.53, cabins=14.88, passenger_density=36.99, crew=19.1, cru
ise_cat=1.0),
 Row(Ship_name='Destiny', Cruise_line='Carnival', Age=17, Tonnage=101.353, pass
engers=26.42, length=8.92, cabins=13.21, passenger_density=38.36, crew=10.0, cr
uise_cat=1.0)]
```

In [11]:
```
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
```

In [12]:
```
indexed.columns
```

Out[12]:
```
['Ship_name',
 'Cruise_line',
 'Age',
 'Tonnage',
 'passengers',
 'length',
 'cabins',
 'passenger_density',
 'crew',
 'cruise_cat']
```

In [13]:
```
assembler = VectorAssembler(
  inputCols=['Age',
             'Tonnage',
             'passengers',
             'length',
             'cabins',
             'passenger_density',
             'cruise_cat'],
    outputCol="features")
```

In [14]:
```
output = assembler.transform(indexed)
```

In [15]:
```
output.select("features", "crew").show(10)
```

```
+--------------------+----+
|            features|crew|
+--------------------+----+
|[6.0,30.276999999...|3.55|
|[6.0,30.276999999...|3.55|
|[26.0,47.262,14.8...| 6.7|
|[11.0,110.0,29.74...|19.1|
|[17.0,101.353,26....|10.0|
|[22.0,70.367,20.5...| 9.2|
|[15.0,70.367,20.5...| 9.2|
|[23.0,70.367,20.5...| 9.2|
|[19.0,70.367,20.5...| 9.2|
|[6.0,110.23899999...|11.5|
+--------------------+----+
only showing top 10 rows
```

In [16]:
```
final_data = output.select("features", "crew")
```

In [17]:
```
train_data, test_data = final_data.randomSplit([0.7,0.3])
```

In [18]:
```
from pyspark.ml.regression import LinearRegression
# Create a Linear Regression Model object
lr = LinearRegression(labelCol='crew')
```

In [19]:
```
# Fit the model to the data and call this model lrModel
lrModel = lr.fit(train_data)
```

In [20]:
```
# Print the coefficients and intercept for linear regression
print("Coefficients: {} Intercept: {}".format(lrModel.coefficients,lrModel.inter
```

```
Coefficients: [-0.02340041799956211,-0.018165101885899433,-0.19748976497916407,
0.4825172523928918,1.161026754003244,3.4306930931576767e-06,0.05894759611146288
5] Intercept: -1.4092287425862482
```

In [21]:
```
test_results = lrModel.evaluate(test_data)
```

In [22]:
```
print("RMSE: {}".format(test_results.rootMeanSquaredError))
print("MSE: {}".format(test_results.meanSquaredError))
print("R2: {}".format(test_results.r2))
```

```
RMSE: 0.9899704901750821
MSE: 0.9800415714174923
R2: 0.9332254104855784
```

In [23]:
```
# R2 of 0.86 is pretty good, Let's check the data a little closer
from pyspark.sql.functions import corr
```

In [24]:
```
df.select(corr('crew','passengers')).show()
```

```
+----------------------+
|corr(crew, passengers)|
+----------------------+
|    0.9152341306065384|
+----------------------+
```

In [25]:
```
df.select(corr('crew','cabins')).show()
```

```
+------------------+
|corr(crew, cabins)|
+------------------+
|0.9508226063578497|
+------------------+
```

Okay, so maybe it does make sense! Well that is good news for us, this is information we can bring to the company!

Hope you enjoyed your first consulting gig!

Chapter 7: Logistic Regression - Pipeline

Ex2: Titanic

Dataset 'titanic.csv'.

Requirement:

- Read data
- Pre-process data.
- With some information: 'Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked' => build a model (use Pipeline) to predict if a passenger on Titanic 'Survived' or not
- Estimate this model.

```
In [1]: import findspark
        findspark.init()

In [2]: from pyspark.sql import SparkSession

In [3]: spark = SparkSession.builder.appName('myproj').getOrCreate()

In [4]: data = spark.read.csv('titanic.csv',inferSchema=True,header=True)

In [5]: data.count()

Out[5]: 891

In [6]: data.printSchema()

        root
         |-- PassengerId: integer (nullable = true)
         |-- Survived: integer (nullable = true)
         |-- Pclass: integer (nullable = true)
         |-- Name: string (nullable = true)
         |-- Sex: string (nullable = true)
         |-- Age: double (nullable = true)
         |-- SibSp: integer (nullable = true)
         |-- Parch: integer (nullable = true)
         |-- Ticket: string (nullable = true)
         |-- Fare: double (nullable = true)
         |-- Cabin: string (nullable = true)
         |-- Embarked: string (nullable = true)
```

In [7]:
```
data.columns
```

Out[7]:
```
['PassengerId',
 'Survived',
 'Pclass',
 'Name',
 'Sex',
 'Age',
 'SibSp',
 'Parch',
 'Ticket',
 'Fare',
 'Cabin',
 'Embarked']
```

In [8]:
```
my_cols = data.select(['Survived',
 'Pclass',
 'Sex',
 'Age',
 'SibSp',
 'Parch',
 'Fare',
 'Embarked'])
```

In [9]:
```
my_final_data = my_cols.na.drop()
```

Working with Categorical Columns

Let's break this down into multiple steps to make it all clear.

In [10]:
```
from pyspark.ml.feature import (VectorAssembler,VectorIndexer,
                                OneHotEncoder,StringIndexer)
```

In [11]:
```
gender_indexer = StringIndexer(inputCol='Sex',
                               outputCol='SexIndex')
gender_encoder = OneHotEncoder(inputCol='SexIndex',
                               outputCol='SexVec')
```

In [12]:
```
embark_indexer = StringIndexer(inputCol='Embarked',
                               outputCol='EmbarkIndex')
embark_encoder = OneHotEncoder(inputCol='EmbarkIndex',
                               outputCol='EmbarkVec')
```

In [13]:
```
assembler = VectorAssembler(inputCols=['Pclass',
 'SexVec',
 'Age',
 'SibSp',
 'Parch',
 'Fare',
 'EmbarkVec'],outputCol='features')
```

```
In [14]: from pyspark.ml.classification import LogisticRegression
```

Pipelines

Let's see an example of how to use pipelines (we'll get a lot more practice with these later!)

```
In [15]: from pyspark.ml import Pipeline

In [16]: log_reg_titanic = LogisticRegression(featuresCol='features',
                                               labelCol='Survived')

In [17]: pipeline = Pipeline(stages=[gender_indexer,embark_indexer,
                                     gender_encoder,embark_encoder,
                                     assembler,log_reg_titanic])

In [18]: train_titanic_data, test_titanic_data = my_final_data.randomSplit([0.7,.3])

In [19]: fit_model = pipeline.fit(train_titanic_data)

In [20]: results = fit_model.transform(test_titanic_data)

In [21]: from pyspark.ml.evaluation import BinaryClassificationEvaluator

In [22]: my_eval = BinaryClassificationEvaluator(rawPredictionCol='prediction',
                                                 labelCol='Survived')
```

In [23]:
```
results.select('Survived','prediction').show()
```

```
+--------+----------+
|Survived|prediction|
+--------+----------+
|       0|       1.0|
|       0|       1.0|
|       0|       1.0|
|       0|       1.0|
|       0|       1.0|
|       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|       1.0|
|       0|       0.0|
|       0|       0.0|
|       0|       0.0|
|       0|       0.0|
|       0|       0.0|
|       0|       1.0|
+--------+----------+
only showing top 20 rows
```

In [24]:
```
AUC = my_eval.evaluate(results)
```

In [25]:
```
AUC
```

Out[25]:
```
0.7974789915966386
```