

# 7082 CEM Big Data Management and Data Visualisation

In [1]:

```
! pip install pyspark
```

Collecting pyspark

Downloading

<https://files.pythonhosted.org/packages/89/db/e18cfd78e408de957821ec5ca56de1250645b05f8523d169803d8df35a64/pyspark-3.1.2.tar.gz> (212.4MB)

 212.4MB 65kB/s

Collecting py4j==0.10.9

Downloading

<https://files.pythonhosted.org/packages/9e/b6/6a4fb90cd235dc8e265a6a2067f2a2c99f0d91787f06aca4bcf7c23f3f80/py4j-0.10.9-py2.py3-none-any.whl> (198kB)

 204kB 15.7MB/s

Building wheels for collected packages: pyspark

Building wheel for pyspark (setup.py) ... done

Created wheel for pyspark: filename=pyspark-3.1.2-py2.py3-none-any.whl size=212880768

sha256=19c7228394cd6215941c37aeeb97fd494688b0d5665558ad14d0d47a30f63c81

Stored in directory: /root/.cache/pip/wheels/40/1b/2c/30f43be2627857ab80062bef1527c0128f7b4070b6b2d02139

Successfully built pyspark

Installing collected packages: py4j, pyspark

Successfully installed py4j-0.10.9 pyspark-3.1.2

In [3]:

```
#Preparing Spark Session
```

```
from pyspark.sql import SparkSession, DataFrame, functions as F
from pyspark.ml.feature import Imputer, StringIndexer, VectorIndexer, VectorAssembler, OneHotEncoder, PCA, Bucketizer
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml import Pipeline
```

```
import pandas as pd
```

```
import pandas_profiling
```

```
import os
```

```
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

---

In [6]:

```
#Mounting Google Drive for loading data
```

```
from google.colab import drive
```

```
drive.mount('/content/drive')
```

Mounted at /content/drive

---

In [7]:

```
#Creating Spark Session
```

```
spark = SparkSession.builder.appName("Titanic-Dataset").config('spark.driver.memory','15g').getOrCreate()
```

```
spark #Details of Spark
```

Out[7]:

SparkContext

[Spark UI](#)

Version v3.1.2 Master local[\*] AppName Titanic-Dataset

In [121]:

```
#Loading titanic Train Data
df_train = spark.read.csv('/content/drive/MyDrive/Colab Notebooks/7082-CEM Big Data Management and Data
Visualisation/train.csv', inferSchema = True, header = True)
df_train.show(10)
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen ...	male	22.0	1	0	A/5 21171	7.25	null	S
2	1	1	Cumings, Mrs. Joh...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. ...	female	26.0	0	0	STON/O2. 3101282	7.925	null	S
4	1	1	Futrelle, Mrs. Ja...	female	35.0	1	0	113803	53.1	C123	S
5	0	3	Allen, Mr. Willia...	male	35.0	0	0	373450	8.05	null	S
6	0	3	Moran, Mr. James	male	null	0	0	330877	8.4583	null	Q
7	0	1	McCarthy, Mr. Tim...	male	54.0	0	0	17463	51.8625	E46	S
8	0	3	Palsson, Master. ...	male	2.0	3	1	349909	21.075	null	S
9	1	3	Johnson, Mrs. Osc...	female	27.0	0	2	347742	11.1333	null	S
10	1	2	Nasser, Mrs. Nich...	female	14.0	1	0	237736	30.0708	null	C

only showing top 10 rows

In [10]:

```
#Printing the datatypes of all columns
df_train.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 [59]:

```
#Selecting only particular features of Dataset
df_train.select("Survived", "Pclass", "Embarked").show()
```

+-----+-----+-----+			
Survived Pclass Embarked			
+-----+-----+-----+			
	0	3	S
	1	1	C
	1	3	S
	1	1	S
	0	3	S
	0	3	Q
	0	1	S
	0	3	S
	1	3	S
	1	2	C
	1	3	S
	1	1	S
	0	3	S
	0	3	S
	0	3	S
	1	2	S
	0	3	Q
	1	2	S

	0	3	S
	1	3	C
+-----+-----+-----+			

only showing top 20 rows

In [65]:

```
#Predicting number of Passengers Survived
#Survived 1 # Not Survived 0
df_train.groupby("Survived").count().show()
```

+-----+-----+			
	Survived	count	
+-----+-----+			
	1	342	
	0	549	
+-----+-----+			

In [66]:

```
#Survival Rate based on Gender
#Survived 1 # Not Survived 0
df_train.groupby("Sex","Survived").count().show()
```

+-----+-----+-----+			
	Sex	Survived	count
+-----+-----+-----+			
	male	0	468
	female	1	233
	female	0	81
	male	1	109

+-----+-----+-----+

In [13]:

*#Loading Test Data*

```
df_test = spark.read.csv('/content/drive/MyDrive/Colab Notebooks/7082-CEM Big Data Management and Data  
Visualisation/test.csv', inferSchema = True, header = True)  
df_test.show(10)
```

PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	null	Q
893	3	Wilkes, Mrs. Jame...	female	47.0	1	0	363272	7.0	null	S
894	2	Myles, Mr. Thomas...	male	62.0	0	0	240276	9.6875	null	Q
895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	null	S
896	3	Hirvonen, Mrs. Al...	female	22.0	1	1	3101298	12.2875	null	S
897	3	Svensson, Mr. Joh...	male	14.0	0	0	7538	9.225	null	S
898	3	Connolly, Miss. Kate	female	30.0	0	0	330972	7.6292	null	Q
899	2	Caldwell, Mr. Alb...	male	26.0	1	1	248738	29.0	null	S
900	3	Abraham, Mrs. Jos...	female	18.0	0	0	2657	7.2292	null	C
901	3	Davies, Mr. John ...	male	21.0	2	0	A/4 48871	24.15	null	S

only showing top 10 rows

In [14]:

*#Printing the datatypes of all columns*

```
df_test.printSchema()
```

```
-- PassengerId: 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 [17]:

## #Descriptive Statistics

```
df_train.describe().show()
```

Titanic Data Summary										
summary	PassengerId			Survived	Pclass	Name	Sex	Age	SibSp	Parch
Ticket	Fare	Cabin	Embarked							
count	891		891	891	891	891	714	891	891	891
891  204	889									
mean	446.0	0.3838383838383838	2.308641975308642			null	null			
29.69911764705882	0.5230078563411896	0.38159371492704824	2.60318.54916792738	32.2042079685746	null	null				
stddev	257.3538420152301	0.48659245426485753	0.8360712409770491			null				
null 14.526497332334035	1.1027434322934315	0.8060572211299488	471609.26868834975	49.69342859718089	null	null				
min	1	0	1	"Andersson, Mr. A..."	female	0.42	0	0	110152	
0.0  A10	C									
max	891	1	3	van Melkebeke, Mr..."	male	80.0	8	6	WE/P	
5735	512.3292	T	S							

# Data Preparation¶

In [123]:

```
from pyspark.sql.functions import mean,col,split, col, regexp_extract, when, lit
df_train = df_train.withColumn("Initial",regexp_extract(col("Name"),"([A-Za-z]+)\.",1))
```

In [124]:

```
df_train.show()
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Initial
1	0	3	Braund, Mr. Owen ...	male	22.0	1	0	A/5 21171	7.25	null	S	Mr
2	1	1	Cumings, Mrs. Joh...	female	38.0	1	0	PC 17599	71.2833	C85	C	Mrs
3	1	3	Heikkinen, Miss. ...	female	26.0	0	0	STON/O2. 3101282	7.925	null	S	Miss
4	1	1	Futrelle, Mrs. Ja...	female	35.0	1	0	113803	53.1	C123	S	Mrs
5	0	3	Allen, Mr. Willia...	male	35.0	0	0	373450	8.05	null	S	Mr
6	0	3	Moran, Mr. James	male	null	0	0	330877	8.4583	null	Q	Mr
7	0	1	McCarthy, Mr. Tim...	male	54.0	0	0	17463	51.8625	E46	S	Mr
8	0	3	Palsson, Master. ...	male	2.0	3	1	349909	21.075	null	S	Master
9	1	3	Johnson, Mrs. Osc...	female	27.0	0	2	347742	11.1333	null	S	Mrs
10	1	2	Nasser, Mrs. Nich...	female	14.0	1	0	237736	30.0708	null	C	Mrs
11	1	3	Sandstrom, Miss. ...	female	4.0	1	1	PP 9549	16.7	G6	S	Miss
12	1	1	Bonnell, Miss. El...	female	58.0	0	0	113783	26.55	C103	S	Miss
13	0	3	Saunderscock, Mr. ...	male	20.0	0	0	A/5. 2151	8.05	null	S	Mr
14	0	3	Andersson, Mr. An...	male	39.0	1	5	347082	31.275	null	S	Mr
15	0	3	Vestrom, Miss. Hu...	female	14.0	0	0	350406	7.8542	null	S	Miss
16	1	2	Hewlett, Mrs. (Ma...	female	55.0	0	0	248706	16.0	null	S	Mrs



	17	0	3 Rice, Master. Eugene	male	2.0	4	1	382652	29.125	null	Q	Master
	18	1	2 Williams, Mr. Cha...	male	null	0	0	244373	13.0	null	S	Mr
	19	0	3 Vander Planke, Mr...	female	31.0	1	0	345763	18.0	null	S	Mrs
	20	1	3 Masselmani, Mrs. ...	female	null	0	0	2649	7.225	null	C	Mrs

+-----+-----+-----+-----+-----+-----+-----+-----+-----+

only showing top 20 rows

In [125]:

```
df_train.select("Initial").distinct().show()
```

+-----+
Initial
+-----+
Don
Miss
Countess
Col
Rev
Lady
Master
Mme
Capt
Mr
Dr
Mrs
Sir
Jonkheer
Mlle
Major
Ms
+-----+

In [126]:

```
df_train = df_train.replace(['Mlle','Mme', 'Ms', 'Dr','Major','Lady','Countess','Jonkheer','Col','Rev','Capt','Sir','Don'],
```

```
['Miss','Miss','Miss','Mr','Mr', 'Mrs', 'Mrs', 'Other', 'Other','Other','Mr','Mr','Mr'])
```

---

In [98]:

```
df_train.select("Initial").distinct().show()
```

```
+-----+
|Initial|
+-----+
|  Miss|
|  Other|
| Master|
|   Mr  |
|   Mrs |
+-----+
```

---

In [99]:

```
#Grouping Initial by Age
```

```
df_train.groupby('Initial').avg('Age').collect()
```

Out[99]:

```
[Row(Initial='Miss', avg(Age)=21.86),
 Row(Initial='Other', avg(Age)=45.888888888888886),
 Row(Initial='Master', avg(Age)=4.574166666666667),
 Row(Initial='Mr', avg(Age)=32.73960880195599),
 Row(Initial='Mrs', avg(Age)=35.981818181818184)]
```

---

In [100]:

```
df_train = df_train.withColumn("Age", when((df_train["Initial"] == "Miss") & (df_train["Age"].isNull()), 22).otherwise(df_train["Age"]))
df_train = df_train.withColumn("Age", when((df_train["Initial"] == "Other") & (df_train["Age"].isNull()), 46).otherwise(df_train["Age"]))
df_train = df_train.withColumn("Age", when((df_train["Initial"] == "Master") & (df_train["Age"].isNull()), 5).otherwise(df_train["Age"]))
df_train = df_train.withColumn("Age", when((df_train["Initial"] == "Mr") & (df_train["Age"].isNull()), 33).otherwise(df_train["Age"]))
df_train = df_train.withColumn("Age", when((df_train["Initial"] == "Mrs") & (df_train["Age"].isNull()), 36).otherwise(df_train["Age"]))
```

In [101]:

```
df_train.filter(df_train.Age==46).select("Initial").show()
```

+-----+	
Initial	
+-----+	
	Mr
	Mr
	Mr
+-----+	

In [102]:

```
df_train.groupBy("Embarked").count().show()
```

+-----+-----+	
Embarked count	
+-----+-----+	
	Q  77
	null  2
	C  168

---

In [103]:

```
df_train = df_train.na.fill({"Embarked" : 'S'})
```

---

In [104]:

```
df_train = df_train.drop("Cabin")
```

---

In [105]:

```
df_train.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)
|-- Embarked: string (nullable = false)
|-- Initial: string (nullable = true)
```

---

In [106]:

```
df_train = df_train.withColumn("Family_Size",col('SibSp')+col('Parch'))
```

---

In [107]:

```
df_train.groupBy("Family_Size").count().show()
```

```
+-----+-----+
|Family_Size|count|
+-----+-----+
|      1| 161|
|      6|  12|
|      3|  29|
|      5|  22|
|      4|  15|
|      7|   6|
|     10|   7|
|      2| 102|
|      0| 537|
+-----+-----+
```

---

In [108]:

```
df_train =df_train.withColumn('Alone',lit(0))
```

---

In [109]:

```
df_train = df_train.withColumn("Alone",when(df_train["Family_Size"] == 0, 1).otherwise(df_train["Alone"]))
```

---

In [110]:

```
indexers = [StringIndexer(inputCol=column, outputCol=column+"_index").fit(df_train) for column in ["Sex", "Embarked", "Initial"]]
pipeline = Pipeline(stages=indexers)
df_train = pipeline.fit(df_train).transform(df_train)
```

---

In [111]:

*#Dropping Columns which are not needed*

```
df_train = df_train.drop("PassengerId", "Name", "Ticket", "Cabin", "Embarked", "Sex", "Initial")
```

---

In [112]:

*#Combining all the features*

```
feature = VectorAssembler(inputCols=df_train.columns[1:], outputCol="features")
feature_vector= feature.transform(df_train)
```

---

In [113]:

```
(trainingData, testData) = feature_vector.randomSplit([0.8, 0.2], seed = 11)
```

# Machine Learning Algorithms¶

In [130]:

```
#Logistic Regression
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

lr = LogisticRegression(labelCol="Survived", featuresCol="features")
#Training algorithm
lrModel = lr.fit(trainingData)
lr_pred = lrModel.transform(testData)
lr_pred.select("prediction", "Survived", "features").show()
evaluator = MulticlassClassificationEvaluator(labelCol="Survived", predictionCol="prediction", metricName="accuracy")
```

```
+-----+-----+-----+
|prediction|Survived|      features|
+-----+-----+-----+
|    1.0|      0|0|(10,[0,1,4,6,8],[...|
|    1.0|      0|0|(10,[0,1,4,6,8],[...|
|    0.0|      0|0|(10,[0,1,4,6],[1....|
|    0.0|      0|0|(10,[0,1,2,4,5],[...|
|    0.0|      0|0|(10,[0,1,4,6,8],[...|
|    0.0|      0|0|(10,[0,1,2,4,5],[...|
|    0.0|      0|0|(10,[0,1,6],[1.0,...|
|    0.0|      0|0|(10,[0,1,4,6],[1....|
|    0.0|      0|0|(10,[0,1,4,6],[1....|
|    0.0|      0|0|(10,[0,1,4,6],[1....|
|    1.0|      0|0|(10,[0,1,4,6],[1....|
|    0.0|      0|0|(10,[0,1,2,4,5],[...|
|    1.0|      0|0|(10,[0,1,3,4,5],[...|
|    0.0|      0|0|(10,[0,1,2,4,5],[...|
|    0.0|      0|0|(10,[0,1,4,6],[1....|
|    0.0|      0|0|[1.0,58.0,0.0,2.0...|
|    0.0|      0|0|(10,[0,1,4,6],[1....|
|    0.0|      0|0|(10,[0,1,4,6],[1....|
|    0.0|      0|0|(10,[0,1,4,6,8],[...|
```

```
| 0.0| 0|[2.0,19.0,1.0,1.0...|
+-----+-----+-----+
only showing top 20 rows
```

In [131]:

```
#Accuracy of the model
lr_accuracy = evaluator.evaluate(lr_pred)
print("Accuracy of LogisticRegression is = %g"% (lr_accuracy))
print("Test Error of LogisticRegression = %g " % (1.0 - lr_accuracy))
```

```
Accuracy of LogisticRegression is = 0.771277
Test Error of LogisticRegression = 0.228723
```

In [117]:

```
#Decision Tree Classifier
from pyspark.ml.classification import DecisionTreeClassifier
dt = DecisionTreeClassifier(labelCol="Survived", featuresCol="features")
dt_model = dt.fit(trainingData)
dt_prediction = dt_model.transform(testData)
dt_prediction.select("prediction", "Survived", "features").show()
```

+-----+-----+-----+		
prediction	Survived	features
+-----+-----+-----+		
0.0	0	(10,[0,1,4,6,8],[...
0.0	0	(10,[0,1,4,6,8],[...
1.0	0	(10,[0,1,4,6],[1....
0.0	0	(10,[0,1,2,4,5],[...
1.0	0	(10,[0,1,4,6,8],[...
0.0	0	(10,[0,1,2,4,5],[...



```
| 0.0| 0|(10,[0,1,6],[1.0,...|
| 0.0| 0|(10,[0,1,4,6],[1....|
| 0.0| 0|(10,[0,1,4,6],[1....|
| 0.0| 0|(10,[0,1,4,6],[1....|
| 0.0| 0|(10,[0,1,4,6],[1....|
| 0.0| 0|(10,[0,1,2,4,5],[...|
| 0.0| 0|(10,[0,1,3,4,5],[...|
| 0.0| 0|(10,[0,1,2,4,5],[...|
| 0.0| 0|(10,[0,1,4,6],[1....|
| 0.0| 0|[1.0,58.0,0.0,2.0...|
| 0.0| 0|(10,[0,1,4,6],[1....|
| 0.0| 0|(10,[0,1,4,6],[1....|
| 0.0| 0|(10,[0,1,4,6,8],[...|
| 0.0| 0|[2.0,19.0,1.0,1.0...|
```

```
+-----+-----+-----+-----+
```

only showing top 20 rows

---

In [118]:

```
dt_accuracy = evaluator.evaluate(dt_prediction)
print("Accuracy of DecisionTreeClassifier is = %g"% (dt_accuracy))
print("Test Error of DecisionTreeClassifier = %g " % (1.0 - dt_accuracy))
```

```
Accuracy of DecisionTreeClassifier is = 0.819149
Test Error of DecisionTreeClassifier = 0.180851
```