Big Data vs Small Data

FOOTBALL EVENTS

Amir Firdaus - Imran Asyraaf - Yasmin Hannah

Group 1

TABLE OF CONTENTS

Introduction

01

Introduction

The football market is vast and involves many different entities such as players, teams, leagues, and match statistics. All of these entities generate a lot of data, which can be used for various purposes. Some of the data is used internally to make better decisions, while other data is used by the media industry to create better products and attract viewers.

Problem Statements

There are hundreds of games of football played in a season and thousands of games in a year. And each game generates a variety of data and variables from different aspects.

Limited Scalability

- Small data tools may not be able to handle large amounts of data, making it difficult to scale the data flow as the volume of data increases from sources to end user.

Limited data storage and processing capabilities

- Small data tools may not have the capability to store and process large amounts of data in efficient manner, which can cause delays in decision-making.

Methodology

02

Dataset Used

The dataset is called **Football Events** from **Kaggle** that provides a granular view of **9,074 games**, totaling 941,009 events from the biggest **5 European football (soccer) leagues**: England, Spain, Germany, Italy, France from 2011/2012 season to 2016/2017 season.

The dataset is organized in 3 files:

- **events.csv** - Contains event data about each game.
- **ginf.csv** - Contains metadata and market odds about each game.
- **dictionary.txt** - Contains a dictionary with the textual description of each categorical variable coded with integers.

Tools Used

Small Data Tools	Aspect	Big Data Tools
Excel	**Data Storage**	Hadoop & Hive
RStudio	**Data Analysis**	PySpark
Python	**Machine Learning**	PySpark

Comparisons

03

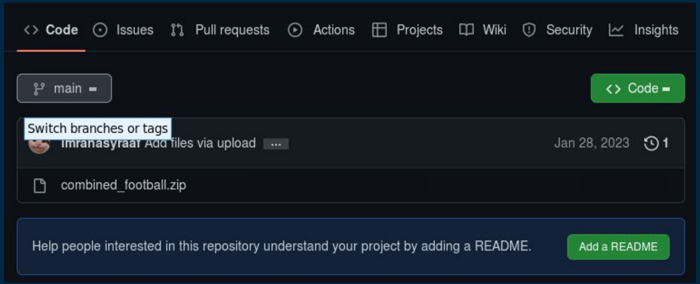
Data Storage [Excel vs Hadoop, Hive]

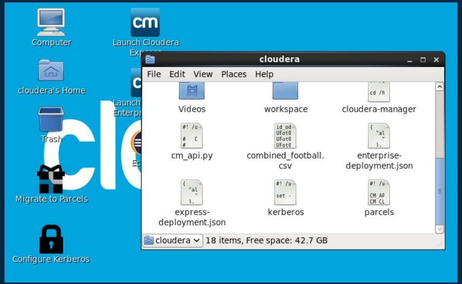
- Ensures the integrity and accuracy of the data. Storing data in a safe and organized manner prevents it from being lost, corrupted, or modified accidentally.

- Allows for easy retrieval and analysis of the data. Properly storing data in a way that is easily searchable and accessible can save time and improve decision-making.

Storing Big Data – Hadoop & Hive

1. Downloading data from resource





2. Make Directory in HDFS

```
205 NodeManager
root@quickstart cloudera]# hadoop fs -mkdir /BigData
root@quickstart cloudera]# hadoop fs -mkdir /BigData/csv
root@quickstart cloudera]#
```

Browse Directory

/

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	root	supergroup	0 B	Sat Jan 28 08:33:07 -0800 2023	0	0 B	BigData
drwxrwxrwx	hdfs	supergroup	0 B	Mon Oct 23 09:15:43 -0700 2017	0	0 B	benchmarks
drwxr-xr-x	hbase	supergroup	0 B	Sat Jan 28 06:31:54 -0800 2023	0	0 B	hbase
drwxr-xr-x	solr	solr	0 B	Mon Oct 23 09:18:01 -0700 2017	0	0 B	solr
drwxrwxrwt	hdfs	supergroup	0 B	Tue Jan 10 18:31:22 -0800 2023	0	0 B	tmp
drwxr-xr-x	hdfs	supergroup	0 B	Mon Oct 23 09:17:33 -0700 2017	0	0 B	user
drwxr-xr-x	hdfs	supergroup	0 B	Mon Oct 23 09:17:24 -0700 2017	0	0 B	var

Hadoop, 2017.

Storing Big Data - Hadoop & Hive

4. Loading data into HDFS

Browse Directory

/BigData/csv Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	181.97 MB	Sat Jan 28 08:41:44 -0800 2023	1	128 MB	combined_football.csv

Hadoop, 2017.

5. Create Hive Table

```
hive> create table combined_football(
    > shot_place int, shot_outcome int, is_goal int, bodypart int, location int, assist_method int, situation int);
OK
Time taken: 1.129 seconds
hive>
```

6. Create simple query

```
hive> select *
    > from football
    > where is_goal = 1;
```

```
Time taken: 0.05 seconds, Fetched: 8439 row(s)
```

7. Result in Hive

NULL	NULL	1	2	NULL	1	NULL
NULL	NULL	1	1	NULL	1	12
NULL	NULL	1	1	NULL	3	NULL
NULL	NULL	1	2	NULL	NULL	NULL
NULL	NULL	1	3	NULL	8	NULL
NULL	NULL	1	1	NULL	3	NULL
NULL	NULL	1	2	NULL	3	NULL
NULL	NULL	1	3	NULL	1	12
NULL	NULL	1	3	NULL	1	12
NULL	NULL	1	1	NULL	8	NULL
NULL	NULL	1	3	NULL	3	NULL
NULL	NULL	1	2	NULL	1	12
NULL	NULL	1	1	NULL	3	NULL
NULL	NULL	1	1	NULL	10	NULL
NULL	NULL	1	5	NULL	1	12

Storing Small Data - Microsoft Excel

shot_place	shot_outcome	is_goal	location	bodypart	assist_method	situation
4	1	1	9	2	1	1
5	1	1	3	1	1	1
4	1	1	13	1	0	3
3	1	1	3	2	0	3
3	1	1	15	1	1	1
12	1	1	3	3	2	1
12	1	1	3	3	2	1
3	1	1	3	3	2	1
4	1	1	13	1	0	1
4	1	1	13	3	2	1
3	1	1	10	1	2	2
13	1	1	15	1	1	1
3	1	1	3	2	1	1
12	1	1	9	1	0	3

- For small data, we only stores it inside Excel file (.csv format)
- Filter features - selecting data that we want without using query.

Data Storing – Comparison Table

Excel	Aspect	Hadoop & Hive
Design for small to medium dataset and has its own limit.	**Data Scale**	Designed for handling large amounts of data, often in the terabytes or petabytes range.
Excel, on the other hand, may become slow when dealing with large data sets.	**Performance**	Hadoop is designed for parallel processing, which makes it faster and more efficient at handling large amounts of data
data can be filtered using the filter function	**Data Extraction**	Hive SQL queries can filter data using a query language with more complexity.
Excel files can be easily shared and accessed by multiple users	**Accessibility**	data stored in Hadoop often requires specialized tools and knowledge to access and analyze

Data Analysis [R vs PySpark]

- To gain deeper understanding and comprehension of the information in the dataset

- Questions that asked:
    - What are the minutes throughout the game that have the highest frequency of events?
    - Which type of events had the most occurrences during the game?
    - Who are the most frequent players and what types of events do they participate in?

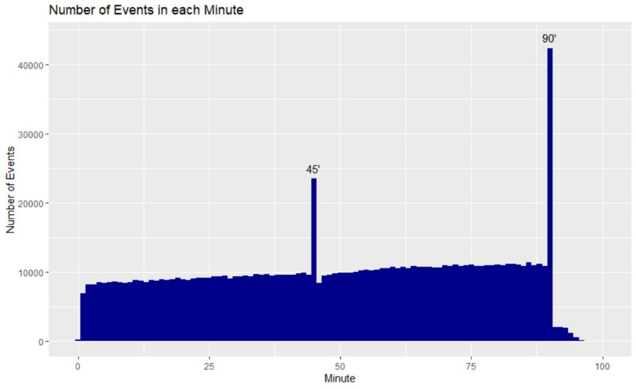
Data Analysis - Dependencies Used

R
- tidyverse
- dplyr
- ggplot2

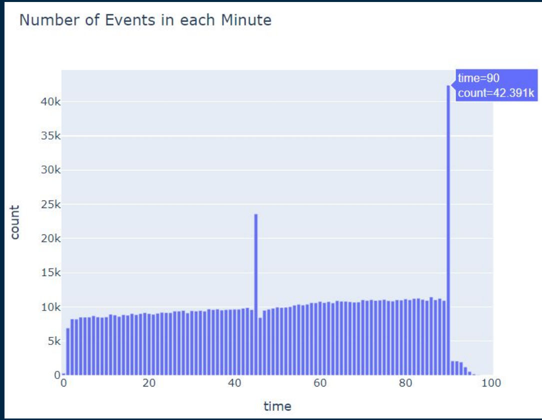
PySpark
- pyspark.sql
- pyspark.sql.types
- Pyspark.sql.functions
  - col
  - udf
- plotly.express
- pandas

Data Analysis - Question 1

What are the minutes with the highest frequency of events?



R



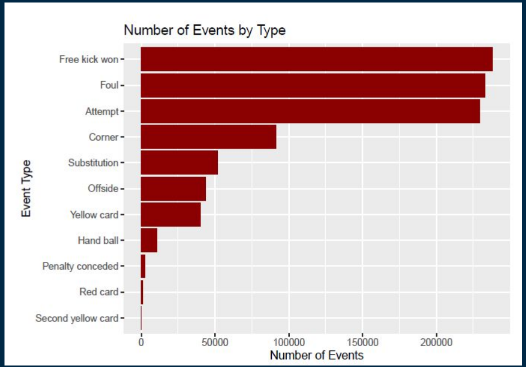
PySpark

These bar charts, illustrate an overall upward trend in the number of events as the minutes progress. Notably, there are two significant peaks present in the plot, specifically at the **45th** and **90th minutes**.

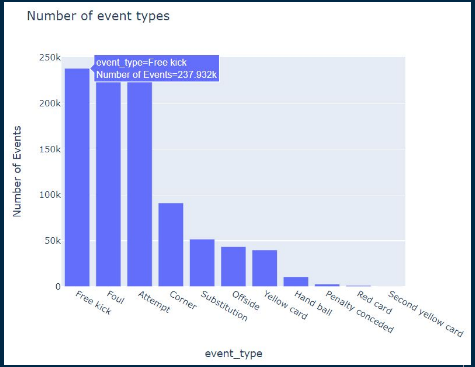
Data Analysis - Question 2

What are the minutes with the highest frequency of events?

event_type	Number of Events
Free kick	237932
Foul	232925
Attempt	229135
Corner	91204
Substitution	51738
Offside	43476
Yellow card	39911
Hand ball	10730
Penalty conceded	2706
Red card	1152
Second yellow card	100



R

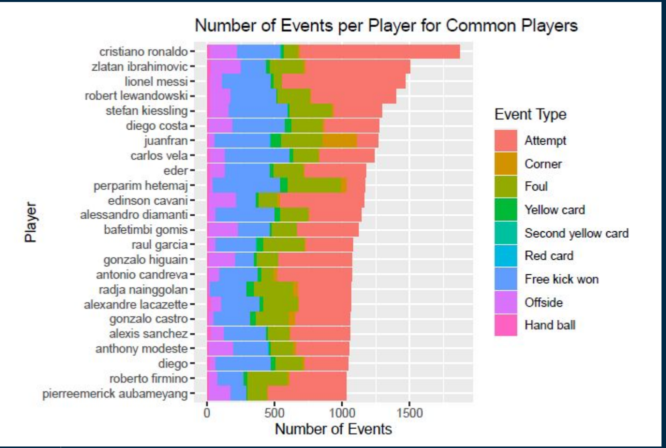


PySpark

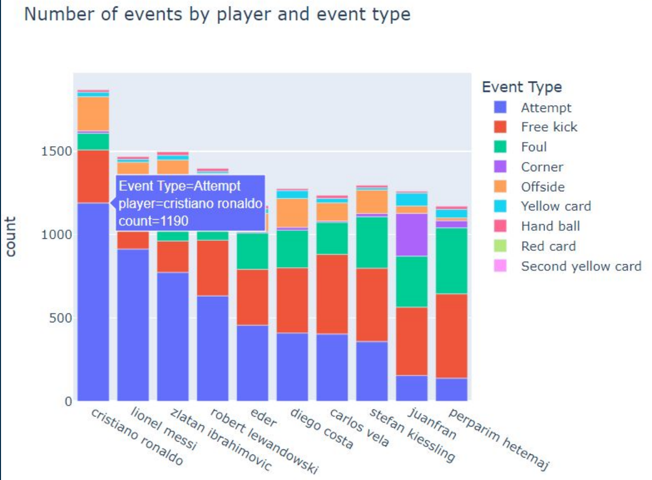
These bar charts display the frequency of different events that occurred during a game, with the majority of the events being **free kicks**, followed by **fouls** and **attempts.** This chart is useful for understanding the types of events that were most common during the game and can be used to identify patterns or trends in the way the game was played.

Data Analysis – Question 3

Who are the most frequent players and what types of events do they participate in?



R



PySpark

These stacked bar charts display the most frequent players in the data.

Typically, these players are world-class forwards who have a high number of attempts

Data Analysis – Overall Comparison

R	Aspect	PySpark
Does not require extensive setup and configuration. All that is needed is to import the necessary libraries for the analysis and visualization tasks.	**Setup and Configuration**	Required a few steps to install and configure the Spark environment on Google Colab. Also need to import necessary libraries used during the analysis and visualizations.
It requires knowledge of the R programming language but there are many resources available for learning and troubleshooting with a bigger and active community of users.	**Ease of use**	It requires the knowledge of Python and SQL, and has a smaller community. Finding and learning resources and tools may be harder.
Provide more advanced and sophisticated packages like ggplot2, lattice, etc	**Visualizations**	Also provide visualization libraries that integrate with Python such as matplotlib, plotly, seaborn, etc.
Not optimized for handling large amounts of data in a distributed environment, and can be slower in terms of loading data into memory compared to big data tools.	**Distributed computing**	Able to process large amounts of data in a distributed environment. Load data into the session much faster.

Machine Learning [Python vs PySpark]

- Multivariate logistic regression to predict a binary classification whether it is a goal or not (1 is goal and 0 is not goal).
- The features used are "side", "shot_place", "location", "assist_method", and "situation".

Machine Learning - Dependencies Used

Python

- For data processing: pandas, numpy
- For machine learning: scikit-learn
- For visualization: matplotlib, sns

PySpark

- For data processing: pyspark.sql
- For machine learning: pyspark.ml
- For visualization: matplotlib, sns

```
#data pre-processing libraries
import pandas as pd
import numpy as np

#machine learning libraries
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import train_test_split
from sklearn import metrics

#visualization libraries
import matplotlib.pyplot as plt
import seaborn as sns
```

Python

```
#data preprocessing
from pyspark.sql import SparkSession
from pyspark.sql.types import *

#machine learning
from pyspark.ml.classification import LogisticRegression
from pyspark.ml import Pipeline
from pyspark.ml.feature import OneHotEncoder, StringIndexer, VectorAssembler
from pyspark.ml.evaluation import BinaryClassificationEvaluator, MulticlassClassificationEvaluator

#visualization
import matplotlib.pyplot as plt
import seaborn as sns
```

PySpark

Machine Learning – Algorithm

```python
#split dataset in features and target variable
feature_cols = ["side", "shot_place", "location", "assist_method", "situation"]
X = df[feature_cols] # Features
y = df.is_goal # Target variable

# split X and y into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=16)
```

```python
# instantiate the model (using the default parameters)
logreg = LogisticRegression(random_state=16)

# fit the model with data
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
```

Python

```python
# Create a list of variables to used
variables = ["side", "shot_place", "location", "assist_method", "situation"]

# Combine all variables into a single feature vector
featureAssemblerlr = VectorAssembler(inputCols = variables, outputCol = "features")
```

```python
#instantiate the model and pipeline
lr = LogisticRegression(featuresCol="features", labelCol="is_goal", regParam=1.0)
pipelineStageslr = [featureAssembler, lr]
pipelinelr = Pipeline(stages=pipelineStages)

# Split dataset into training/test, and create a model from training data
(trainingData, testData) = gamesDf.randomSplit([0.75, 0.25])
lrmodel = pipelinelr.fit(trainingData)
```

PySpark

Although using different libraries, the algorithm steps are still similar where we first chose the columns to use, split into training and testing, then fit into the logistic model.

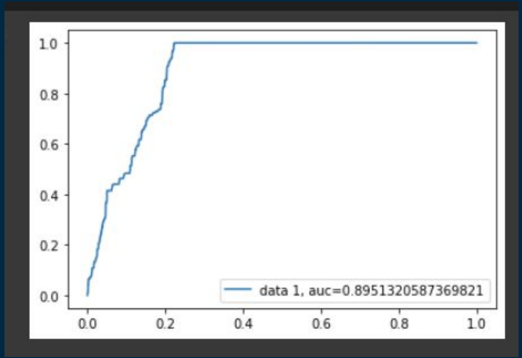
Machine Learning – Evaluation

```
[ ] acc = metrics.accuracy_score(y_test, y_pred)
    print("Accuracy: ", round(acc,2)*100, "%")

    Accuracy:  97.0 %

[ ] y_pred_proba = logreg.predict_proba(X_test)[::,1]
    fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
    auc = metrics.roc_auc_score(y_test, y_pred_proba)
    plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
    plt.legend(loc=4)
    plt.show()
```

Python




```
[70] bcEvaluator = BinaryClassificationEvaluator(labelCol="is_goal", rawPredictionCol="prediction", metricName="areaUnderROC")
     print(f"Area under ROC curve: {bcEvaluator.evaluate(predictionslr)}")

     mcEvaluator = MulticlassClassificationEvaluator(labelCol="is_goal",metricName="accuracy")
     print(f"Accuracy: {mcEvaluator.evaluate(predictionslr)}")

     Area under ROC curve: 0.7780702110316613
     Accuracy: 0.9838811388446232
```

PySpark

Both Python and PySpark have built-in metrics for Area Under Curve (AUC) and Accuracy. From the results, Python Scikit-learn logistic regression has accuracy of 97% and AUC of 0.895 while PySpark MLlib logistic regression has accuracy of 98.4% and AUC of 0.78.

Machine Learning – Comparison Table

Python	Aspect	PySpark
Works very well with small data. Can be faster than PySpark.	**Small Data**	May **introduce unnecessary overhead**, as the distributed computing capabilities of Spark may not be needed for such small datasets.
Has a **wide range of machine learning** libraries such as scikit-learn, Tensorflow and Pytorch, which provides a lot of flexibility.	**Built-in ML support**	Has built-in support for machine learning through its **MLlib library** although it is not as flexible as Python.

Conclusion

04

Limitations & Future Works

Limitations

- Lacking size of the dataset
- Limited representation of end-to-end pipeline

Future Works

- Using real-time data ingestion from data source
- Design and construct pipeline from source to end-user

Summary

The results of this project demonstrate the capabilities of both small and big data tools in storing, analyzing, visualizing, and perform machine learning algorithms on the datasets.

However, as the dataset grows, it may become challenging to handle football events data that generated almost every day using only small data tools such (Excel, R, Python), indicating that big data tools (Hadoop, Hive, PySpark) are more suitable and practical in the long term to solve the scalability and processing capabilities issues arise.

Thank You

From Group 1