Master Executive di II Livello

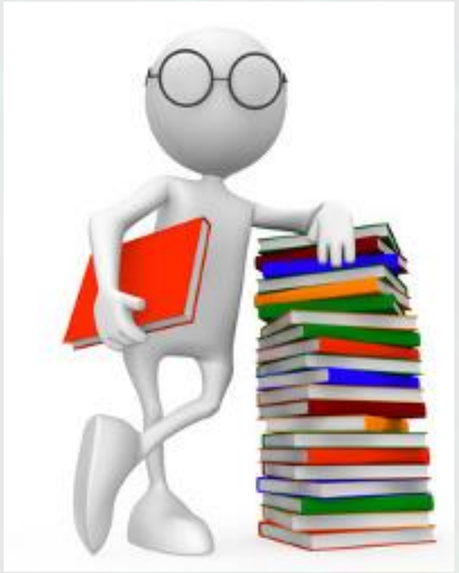
BIG DATA ANALYSIS AND BUSINESS INTELLIGENCE

Vamsi Krishna Varma Gunturi
Data science intern at ISTAT
vamsivarmagunturi@gmail.com

Introduction to Pig

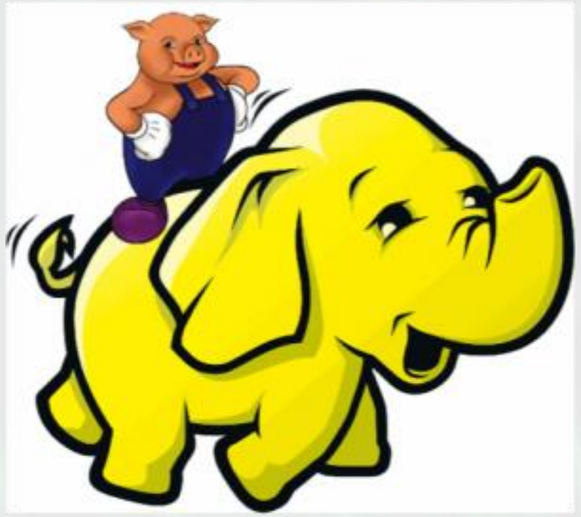
fondazione INUIT TORVERGATA

Topics



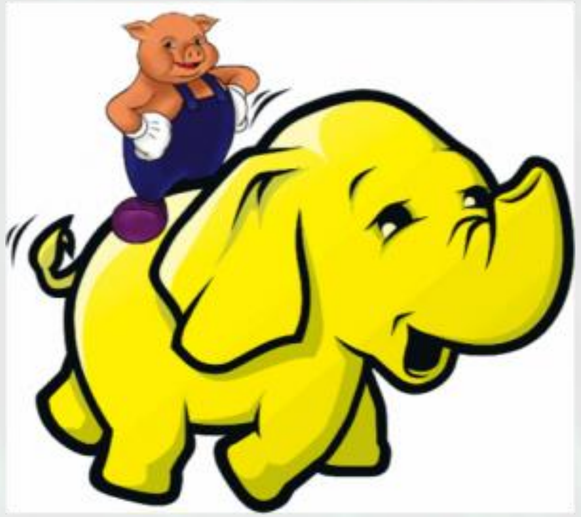
- What is Pig ?
- Why Pig ?
- Where does Pig fit in ?
- MapReduce vs Pig
- Pig vs Hive
- Pig architecture
- Features of Pig
- Pig commands
- Exercises

What is Pig?



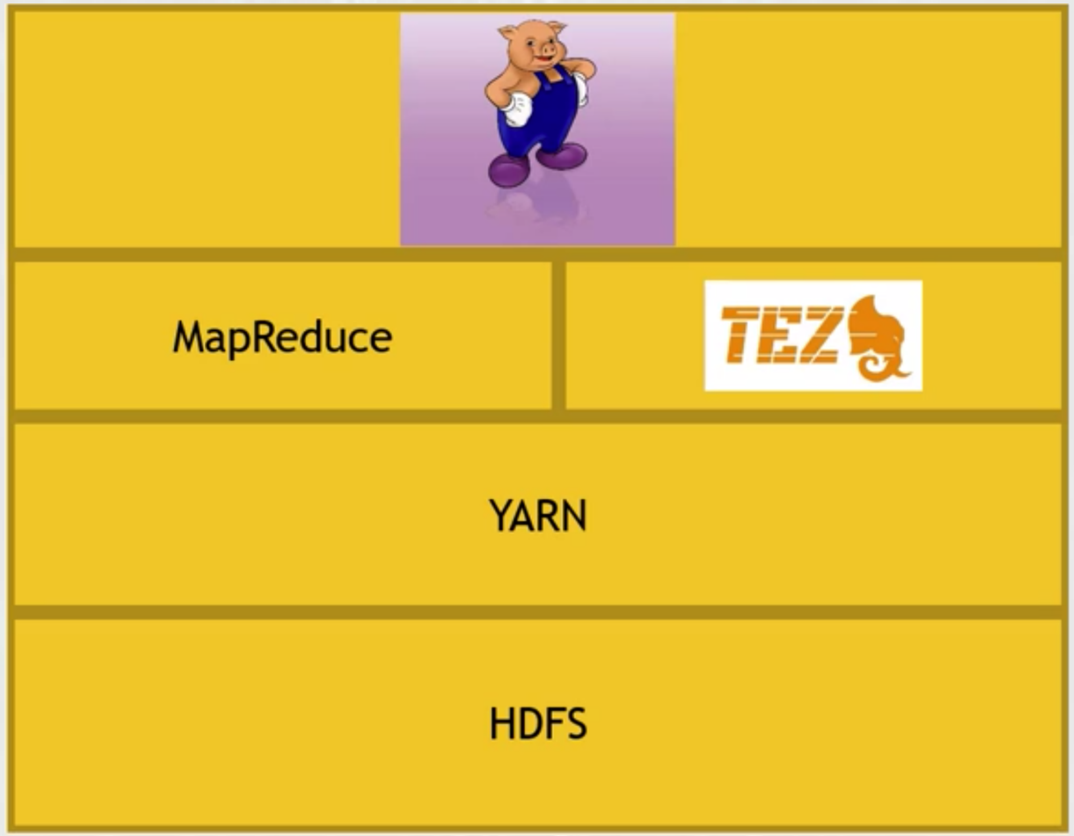
- Apache Pig is a high-level platform for creating programs that run on Apache Hadoop.
- The language for this platform is called Pig Latin.
- Pig can execute its Hadoop jobs in MapReduce, Apache Tez, or Apache Spark.
- Pig Latin abstracts the programming from the Java MapReduce idiom into a notation which makes MapReduce programming high level, similar to that of SQL for relational database management systems.
- Pig Latin can be extended using user-defined functions (UDFs).
- Apache Pig was originally developed at Yahoo Research around 2006 for researchers to have an ad-hoc way of creating and executing MapReduce jobs on very large data sets.

Why Pig?

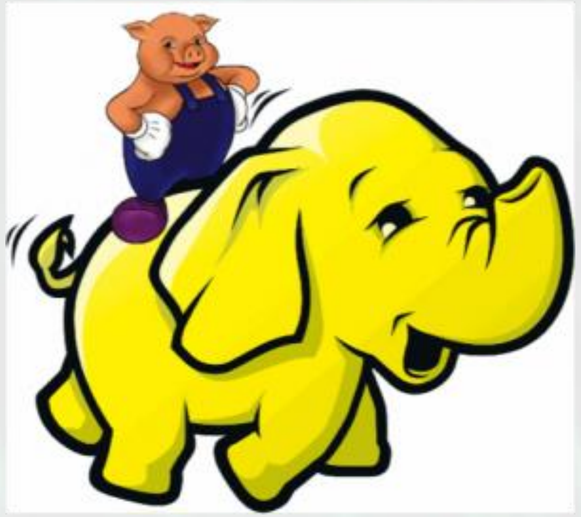


- Writing mappers and reducers by hand takes a long time
- Pig introduces Pig Latin, a scripting language that lets you use SQL-like syntax to define your map and reduce steps.
- Highly extensible with user-defined functions (UDF's).
- Pig Latin structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.

Where does Pig fit in ?

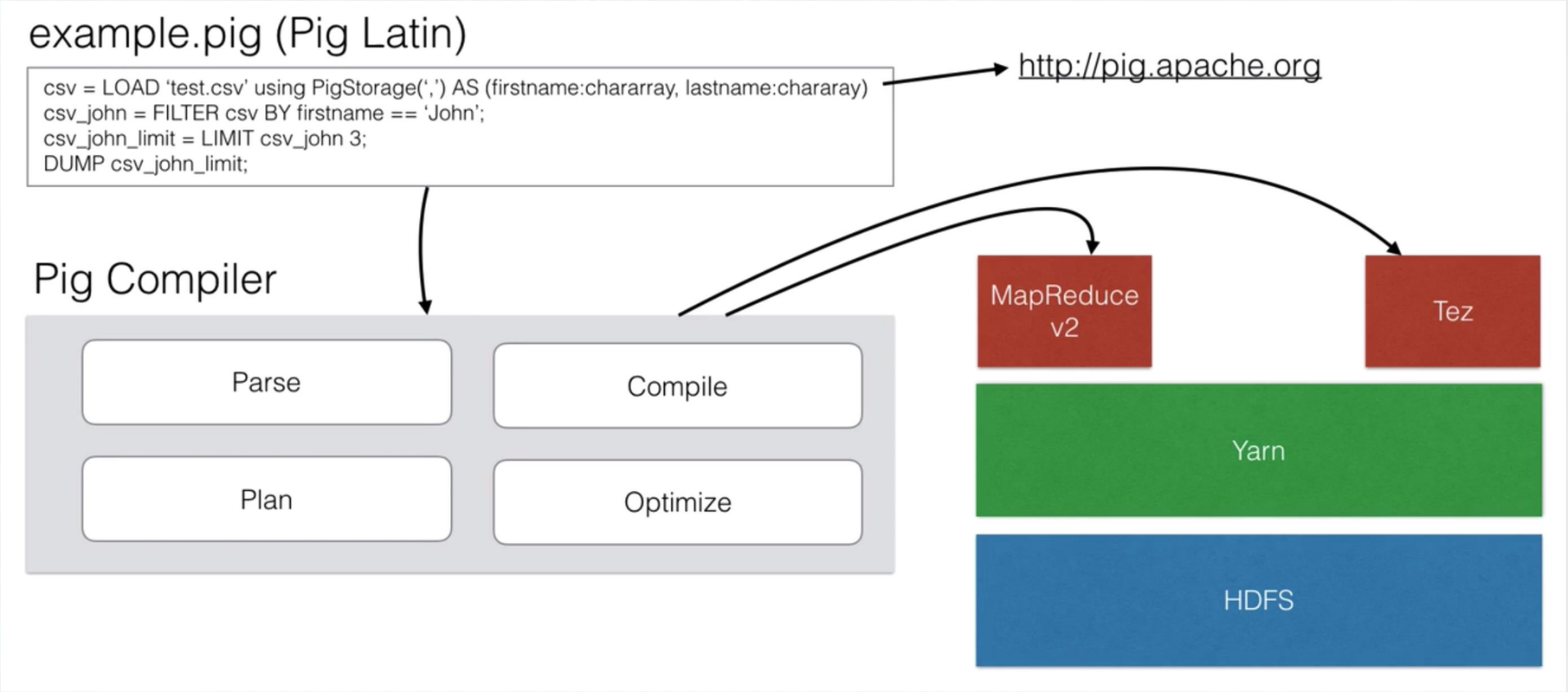


Features

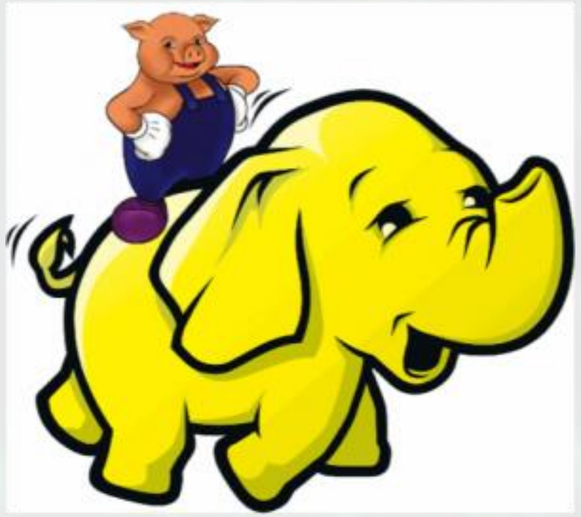


- **Ease of programming**: It is trivial to achieve parallel execution of simple, "embarrassingly parallel" data analysis tasks. Complex tasks comprised of multiple interrelated data transformations are explicitly encoded as data flow sequences, making them easy to write, understand, and maintain.
- **Optimization opportunities:** The way in which tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency.
- **Extensibility:** Users can create their own functions to do special-purpose processing.

Pig Architecture

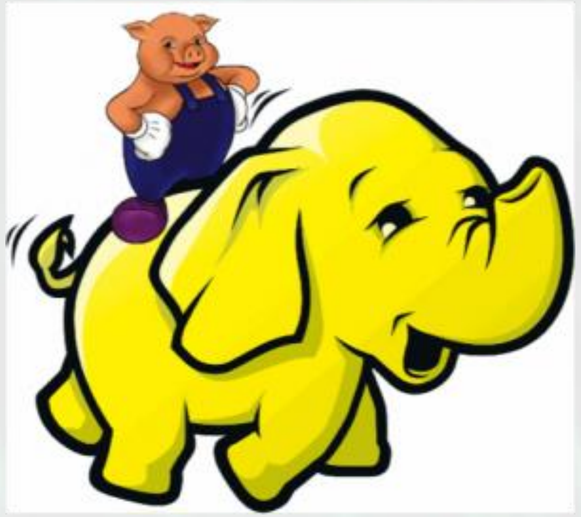


Running Pig



- Grunt
- Script
- Ambari / Hue

Example scripts



- Create a relation with name “ratings” with a given schema – Map operation

```
ratings = LOAD ‘/user/maria_dev/ml-100k/u.data’ AS
(userID: int, movieID: int, rating: int, ratingTime: int)
```

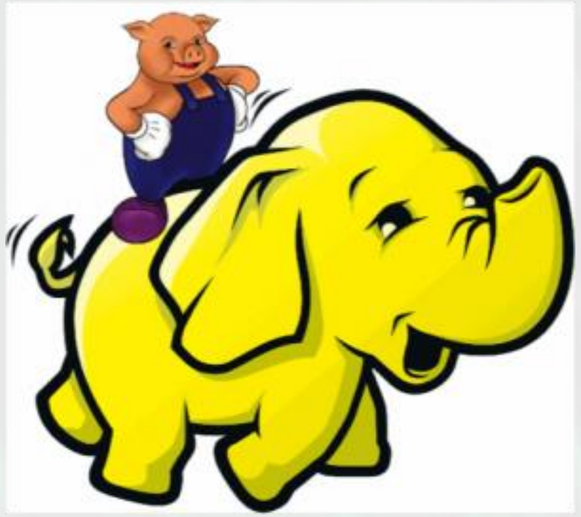
- Use PigStorage if you need a different delimiter –

```
metadata = LOAD ‘/user/maria_dev/ml-100k/u.item’ USING
PigStorage (‘|’) AS (movieID: int, movieTitle: chararray,
releaseDate: chararray, videoRelease: chararray,
imdbLink: chararray );
DUMP metadata;
```

- Create a relation from another relation –

```
nameLookup = FOREACH metadata GENERATE movieID,  movieTitle,
ToUnixTime(ToDate(releaseDate, ‘dd-MM-yyyy’)) AS
releaseTime;
```

Example scripts



- Use Group By – Reduce operation

  ratingsByMovie = GROUP ratings BY movieID;
  DUMP ratingsByMovie;

- Compute average ratings –

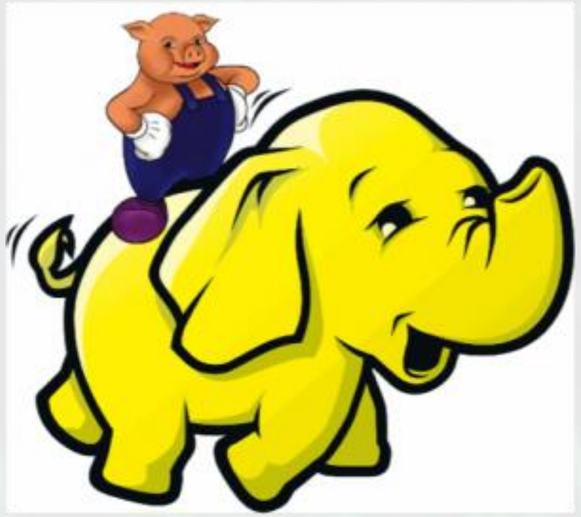
  avgRatings = FOREACH ratingsByMovie GENERATE group AS movieID,
  AVG(ratings.rating) AS avgRating;
  DUMP avgRatings;

- Find the schema of a relation –

  DESCRIBE ratings;

- Filter –
  fiveStarMovies = FILTER avgRatings BY
  avgRating > 4.0;

Example scripts



- JOIN -

```
DESCRIBE fiveStarMovies;
DESCRIBE nameLookup;
fiveStarsWithData = JOIN fiveStarMovies BY movieID, nameLookup BY
                                    movieID;
DESCRIBE fiveStarsWithData;
DUMP fiveStarsWithData;
```

- ORDER BY –

```
oldestFiveStarMovies = ORDER fiveStarsWithData BY
                                    nameLookup::releaseTime;
DUMP oldestFiveStarMovies;
```

Putting it all together..

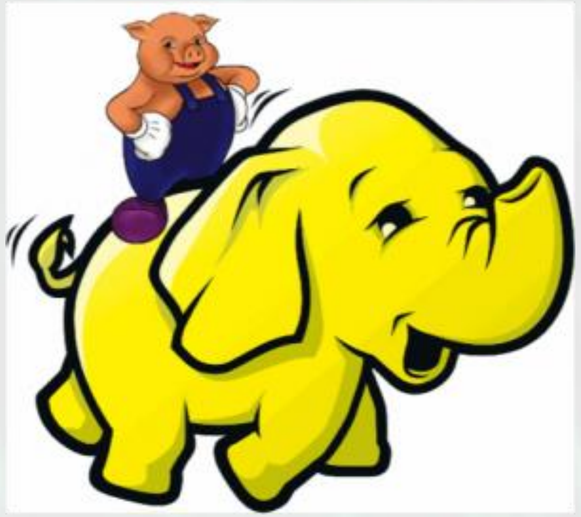
```
ratings = LOAD '/user/maria_dev/ml-100k/u.data' AS (userID:int, movieID:int, rating:int, ratingTime:int);

metadata = LOAD '/user/maria_dev/ml-100k/u.item' USING PigStorage('|')
    AS (movieID:int, movieTitle:chararray, releaseDate:chararray, videoRelease:chararray, imdbLink:chararray);

nameLookup = FOREACH metadata GENERATE movieID, movieTitle,
    ToUnixTime(ToDate(releaseDate, 'dd-MMM-yyyy')) AS releaseTime;

ratingsByMovie = GROUP ratings BY movieID;

avgRatings = FOREACH ratingsByMovie GENERATE group AS movieID, AVG(ratings.rating) AS avgRating;

fiveStarMovies = FILTER avgRatings BY avgRating > 4.0;

fiveStarsWithData = JOIN fiveStarMovies BY movieID, nameLookup BY movieID;

oldestFiveStarMovies = ORDER fiveStarsWithData BY nameLookup::releaseTime;

DUMP oldestFiveStarMovies;
```

Pig Latin Commands

- Operations that you can perform on a relation:

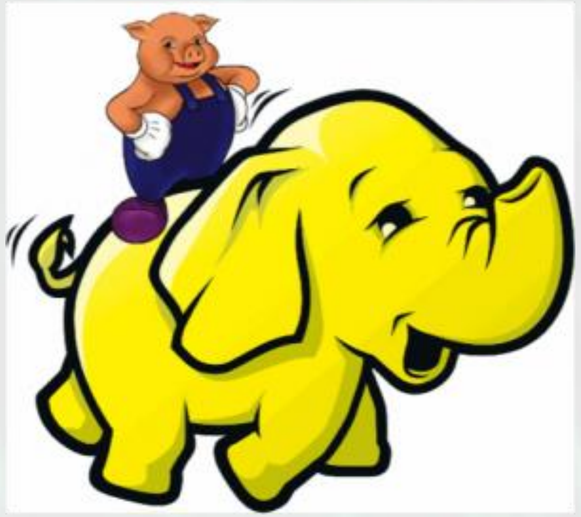
- LOAD STORE DUMP
  - *STORE ratings INTO 'outRatings' USING PigStorage(':');*
- FILTER DISTINCT FOREACH/GENERATE MAPREDUCE STREAM SAMPLE
- JOIN COGROUP GROUP CROSS CUBE
- ORDER RANK LIMIT
- UNION SPLIT

Diagnostics



- DESCRIBE
- EXPLAIN
- ILLUSTRATE

UDF’s



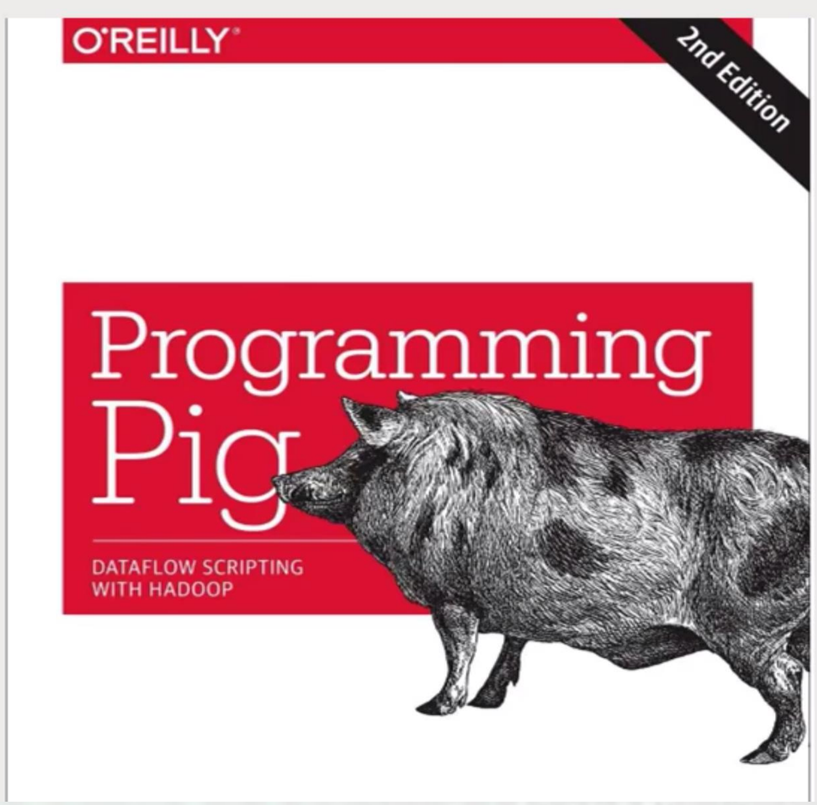
- REGISTER
- DEFINE
- IMPORT

Some other functions and loaders…

- AVG CONCAT COUNT MAX MIN SIZE SUM

- PigStorage
- TextLoader
- JsonLoader
- AvroStorage
- ParquetLoader
- OrcStorage
- HBaseStorage

Learning more…



Master Executive di II Livello

BIG DATA ANALYSIS AND BUSINESS INTELLIGENCE

Vamsi Krishna Varma Gunturi
Data science intern at ISTAT
vamsivarmagunturi@gmail.com

Grazie

fondazione INUIT TORVERGATA