Demand Prediction: EDA

10.28.2017

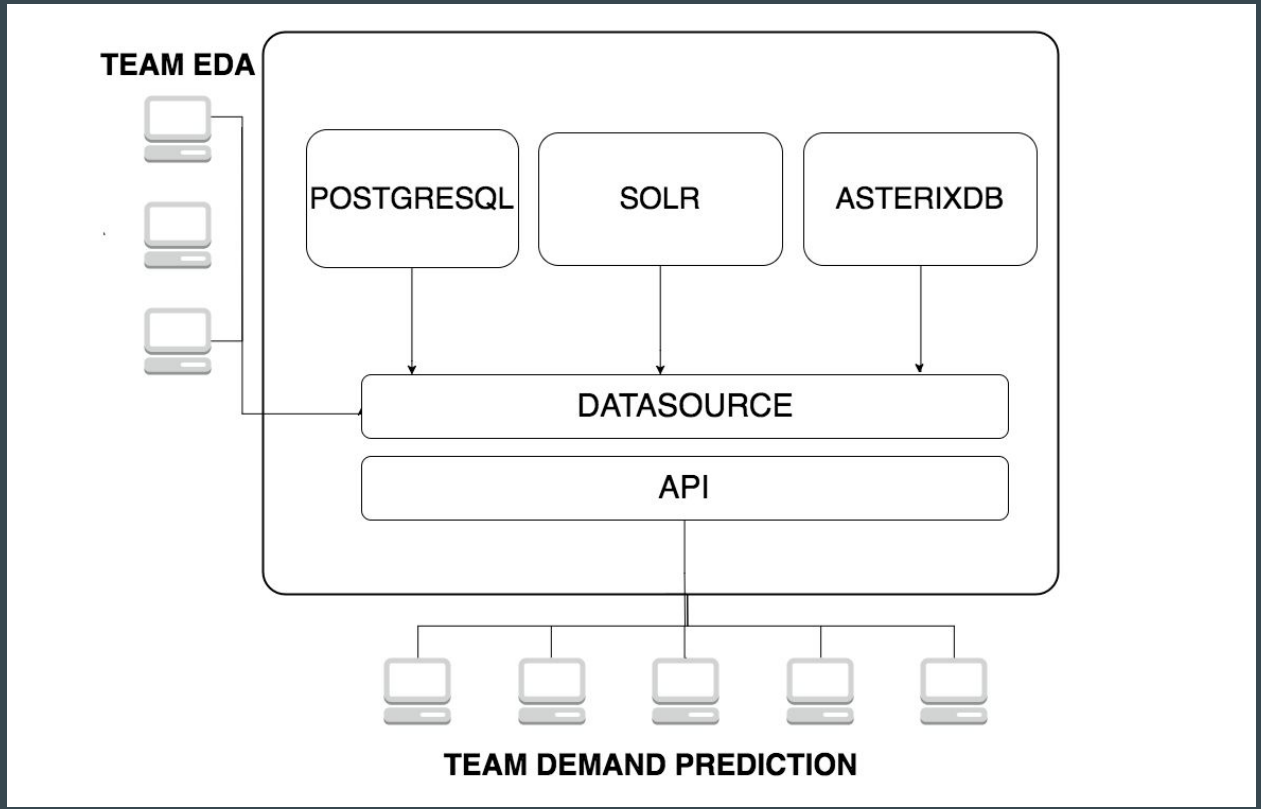
Nirmal Budhathoki
Garrett Cheung
Jillian Jarrett
Toby Moreno
Orysya Stus

Team Update

Second Milestone

- Building Python APIs for Team Demand
  - Proposing a cloud-based architecture for easy and universal access for all teams to build backend services
  - Written and compiled aggregate functions, statistical queries, histograms and exposed them through API available to other teams.
- Compiled (currently supported and future) Functions/API List
- Provide Client wrappers / Sample Visualization (Unit Testing)

Proposed API Cloud Architecture



Linode Server / Data Sources

We have set up linode server (45.79.91.219) for remote access.

- Postgres tables: calendar, campaigns, customers, orderlines, order, products

  *Details for setting up pgadmin and psql from local machine to connect to server will be provided.

- AsterixDB: reviews, classification

  http://45.79.91.219:19001/

- Solr: reviews for bookstore collection

  http://45.79.91.219:8983/solr/#/bookstore/query

API Functionality

Current

/api/service (accepts dynamic params)

/api/covariance

/api/correlation

/api/histogram/groupby/count

Future*

/api/trends/seasonal

/api/sentiment/polarity

/api/clustering/bookcategories

/api/popularity/booksales

* TBD w/ Demand Prediction Team)

Python API Client / Unit Testing

- INPUT
  - Supports JSON input for dynamic parameters
  - Fixed parameters
  - Random Sampling Feature is built-in
- OUTPUT
  - Supports Serialized outputs in JSON (XML*, CSV* format) * if required
  - Deserialized Python Objects (Dictionary, Dataframe) (Sample Visualization)
  - Virtual Integrated Schema (For query builder reference)

API Development Process with Machine Learning and Query Capability/Other Teams and Demo

Machine learning → Define function, inputs, and outputs

Query capability and learning → Optimize data source connections

Integrated schema and justification → Materialize views prior to API calls

```python
@app.route("/api/covariance/<col1>/<col2>")
def covariance(col1, col2):
    """Determine the covariance coefficient between two columns."""

    engine = create_engine('postgresql+psycopg2://postgres@45.79.91.219/MyBookStore')
    conn = engine.connect()

    sql = """
    select covar_samp(%s::int, %s::int)
    from orderlines o, products p
    where o.productid = p.productid
    """ %(col1, col2)

    stmt = text(sql)

    result = conn.execute(stmt)

    for row in result:
        print(row[0])
    conn.close()

    return str(row[0])
```

Sample function code from EDA API.