

ORACLE

Art of Possible with AI & Data Science

Tax Fraud Detection powered by Generative AI, Graph Analytics
and Machine Learning

27/09/23

EMEA Data Science Team

Tax Fraud Briefly

Income Tax Evasion

- Individuals and businesses underreporting their income or inflating deductions to reduce tax liabilities.
- Data Sources: Tax Returns, Wage and Salary Data

VAT/GST Fraud

- Fraudulent schemes involving the collection of value-added tax (VAT) or goods and services tax (GST) but not remitting it to the government.
- Data Sources: Transaction data, invoices, and sales records from businesses
- Taxpayer Records

Corporate Tax Fraud

- Large corporations engaging in aggressive tax planning or transfer pricing schemes to minimize their tax obligations.
- Data Sources: Corporate Financial Statements, Transfer Pricing Documentation

Employment Tax Fraud

- Employers misclassifying employees as independent contractors, thereby avoiding payroll taxes.
- Data Sources: Payroll data

Tax Fraud Briefly

Tax Shelter Abuse

- Exploiting offshore tax shelters or abusive tax shelters to hide income from taxation.
- Data sources: Financial institution data to reveal offshore accounts and financial holdings
- Legal documents

Identity Theft and Refund Fraud

- Criminals using stolen identities to file fraudulent tax returns and claim refunds.
- Data sources: Taxpayer Identification Data,

Sales Tax Fraud

- Retailers underreporting sales or manipulating sales records to evade sales tax.
- Data sources: POS data, inventory records, sales transactions

Cryptocurrency Tax Evasion

- Analyzing blockchain data to trace cryptocurrency transactions and identify tax evasion involving digital assets.
- Cryptocurrency blockchain data, Cryptocurrency Exchange Data



Paysafe:

Real-Time Fraud Detection with Oracle Graph Technologies

“It’s not only about fraud. We can analyze our payments network, our device agent networks, we can feed graph topology and machine learning algorithms. They help us uncover how much more we can do”

Stanka Dalekova

Chief Technical Lead, Paysafe

Paysafe Group

- Online Payments provider, headquartered in Canada
- Annualized transactional volume \$100bn (2020)
- Processing up to 500000 payments per day

Business requirements

- Improve accuracy of fraud detection
- minimize chargebacks from Credit Card providers while not impacting customer experience
- Automate work of analysts in fraud department
- Integrate with operational payment systems (Oracle, MSFT)

Fraud analytics platform based on Oracle Graph

- In-memory analytics engine with millisecond response time enables real-time transaction monitoring
- Integrated interactive visualization of transactions network eliminates manual work of fraud analysts
- Inclusion of account and device-related data in combination with ML improves prediction accuracy
- Seamless integration with payment systems through Oracle GoldenGate

Demo Flow

1. Demo Inspiration & Target Personas
2. How have we achieved this?
 - Business Understanding
 - Data Enhancement with Generative AI
 - Data Exploration
 - Data Preparation (for Graph)
 - Graph Model
 - Graph Metrics
 - Data Preparation (for Machine Learning)
 - Machine Learning Model
 - Business Insights
3. Behind the Scenes
 - Oracle Data Platform
4. Next Steps

The screenshot shows a Jupyter Notebook interface on Oracle Cloud. The notebook has three cells:

- Cell 16:** Enhance data (with GAN). It contains Python code to generate synthetic data from two tables: 'TF_COMPANIES' and 'TF_TRANSACTIONS'. The code uses CTGANSynthesizer to create 100 epochs of synthetic data.
- Cell 17:** Explore data (post-). It shows the result of running the enhancement code, indicating it took 38.7 seconds. The output shows the enhanced data frames for both tables.
- Cell 18:** Explore shape. It displays the shapes of the original and enhanced data frames for both tables.
- Cell 19:** A Cypher query to build a graph. The query matches nodes 'c1' and 'c2' from the 'TF_PG_TAXFRAUD' table and creates edges between them.

Below the notebook is a graph visualization tool showing a network of nodes (blue circles) and edges (green lines). The nodes represent companies, and the edges represent transactions. A legend on the right identifies the vertices and edges.

Demo Inspiration & Target Personas

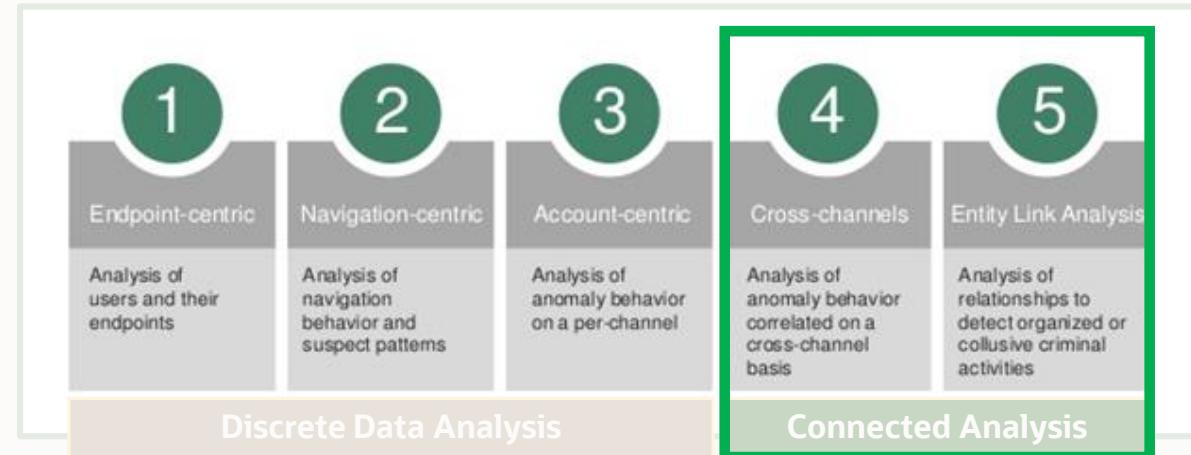
Demo Inspiration

This demo is going to showcase Oracle AI/ML platform capabilities to identify fraudulent behavior in companies through analysis of interactions.

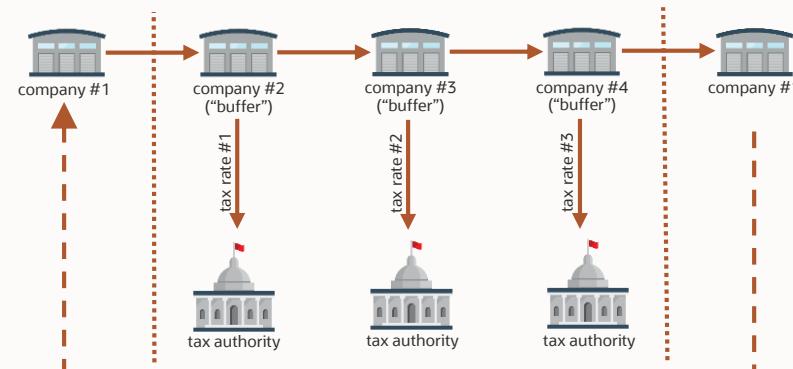
A particular type of behavior investigated includes circular money transfers, which can be indicative of abnormal interactions and can constitute sufficient cause for further analysis.

Objective is uncovering hidden fraud patterns through **graph analysis, machine learning and generative AI** techniques.

Gartner's 5 of Fraud Prevention / Detection



Circular money transfers



Demo Inspiration

With AI driven fraud detection model:

- Tax investigators will be able to identify companies with high fraud risks more effectively
- Tax authorities will better allocate investigator resources
- Government will promote a fair tax system and minimizing revenue losses

Tax fraud poses a significant challenge globally, with estimates suggesting that it costs governments billions of dollars in lost revenue each year. It not only affects public finances but also erodes trust in the tax system.

*HMRC (UK tax collecting authority) estimates that losses to tax fraud amount to **£16 billion** each year. This is nearly half of HMRC's estimate of the tax gap (£34 billion): the difference between the amount of tax HMRC should collect each year and the amount it actually collects.**

*<https://www.nao.org.uk/reports/tackling-tax-fraud-how-hmrc-responds-to-tax-evasion-the-hidden-economy-and-criminal-attacks/>

Target Personas



Fraud Investigator wants to gather evidence to identify tax fraud investigating suspicious activities



Data Scientist wants to use Generative AI, machine learning and graph analytics to develop models and algorithms that can detect fraud patterns in large datasets.



Business Analyst wants to build final dashboards to create visually informative and interactive dashboards that enable fraud investigator to monitor and analyze suspicious activities

How have we achieved this?

Business Understanding



Fraud Investigator wants to understand any cyclic money transfers among companies, as well as importance of said companies.

Cyclic money transactions can be a potential signal of tax fraud because they often indicate an attempt to hide the true nature of financial transactions and income. A measure of importance would also be very useful for prioritization.

Transactions	Companies	Other Potential Data Sources
Transaction ID	Company ID	Geolocation
Company From ID	Company Name	VAT forms/tax returns
Company To ID	...	Whistleblower Reports
Transaction Date		Customs and Import/Export Data
Transaction Amount		Social Media data
...		3 rd party data

tax fraud data sources

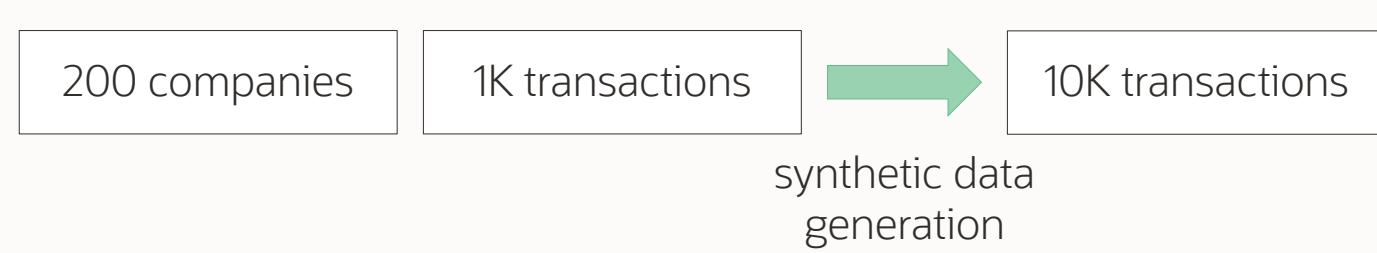
Data Enhancement with Generative AI



Data Scientist assesses available data and sees that existing data volume is very low. She uses Generative AI to increase data volume that helps understanding fraud cases better.

The diagram illustrates the process of generating synthetic data. It starts with two tables: 'Companies' and 'Transactions'. The 'Companies' table contains columns for Company ID, Company Name, and an ellipsis. The 'Transactions' table contains columns for Transaction ID, Company From ID, Company To ID, Transaction Date, Transaction Amount, and an ellipsis. An arrow labeled 'synthetic data generation' points from these initial tables to a final table with the same structure, but with more rows: 200 companies, 1K transactions, and 10K transactions.

Companies		Transactions		Transactions	
Company ID		Transaction ID		Transaction ID	
Company Name		Company From ID		Company From ID	
...		Company To ID		Company To ID	
		Transaction Date		Transaction Date	
		Transaction Amount		Transaction Amount	
		



Data Enhancement with Generative AI

ORACLE Cloud [ODS-02](#) Session remaining: 53min Extend ? Sign Out

File Edit View Run Kernel Git Tabs Settings Help

Launcher 01-enhance.data.with.gan.ipynb Python [conda env:generalml_p38_cpu_v1]

/ work / demo-taxfraud / Name env 01-enhance.data.wit... data.0.xlsx data.1.xlsx

§ Enhance data (with GAN)

```
[16]: 1 # Helpers
2 @run_time_d
3 def get_enhanced_data(label, df, rows):
4     rows = rows - df.shape[0]
5     metadata = SingleTableMetadata()
6     metadata.detect_from_dataframe(data=df)
7     synthesizer = CTGANSynthesizer(metadata, epochs=100)
8     synthesizer.fit(df)
9     df_new = synthesizer.sample(num_rows=rows)
10    df_enh = pandas.concat([df, df_new])
11    df_enh.index = range(1, df_enh.shape[0]+1)
12    df_enh['TRANSACTIONID'] = 'T6438' + (df_enh.index.astype(str).str.zfill(6))
13    return df_enh
```

```
[17]: 1 # Initialize
2 data_1 = {}
3 # Run
4 data_1['TF_COMPANIES'] = data_0['TF_COMPANIES']
5 data_1['TF_TRANSACTIONS'] = get_enhanced_data('TF_TRANSACTIONS', data_0['TF_TRANSACTIONS'], 10000)
```

Running get_enhanced_data(TF_TRANSACTIONS) ... Done (38.7 s)

§ Explore data (post-)

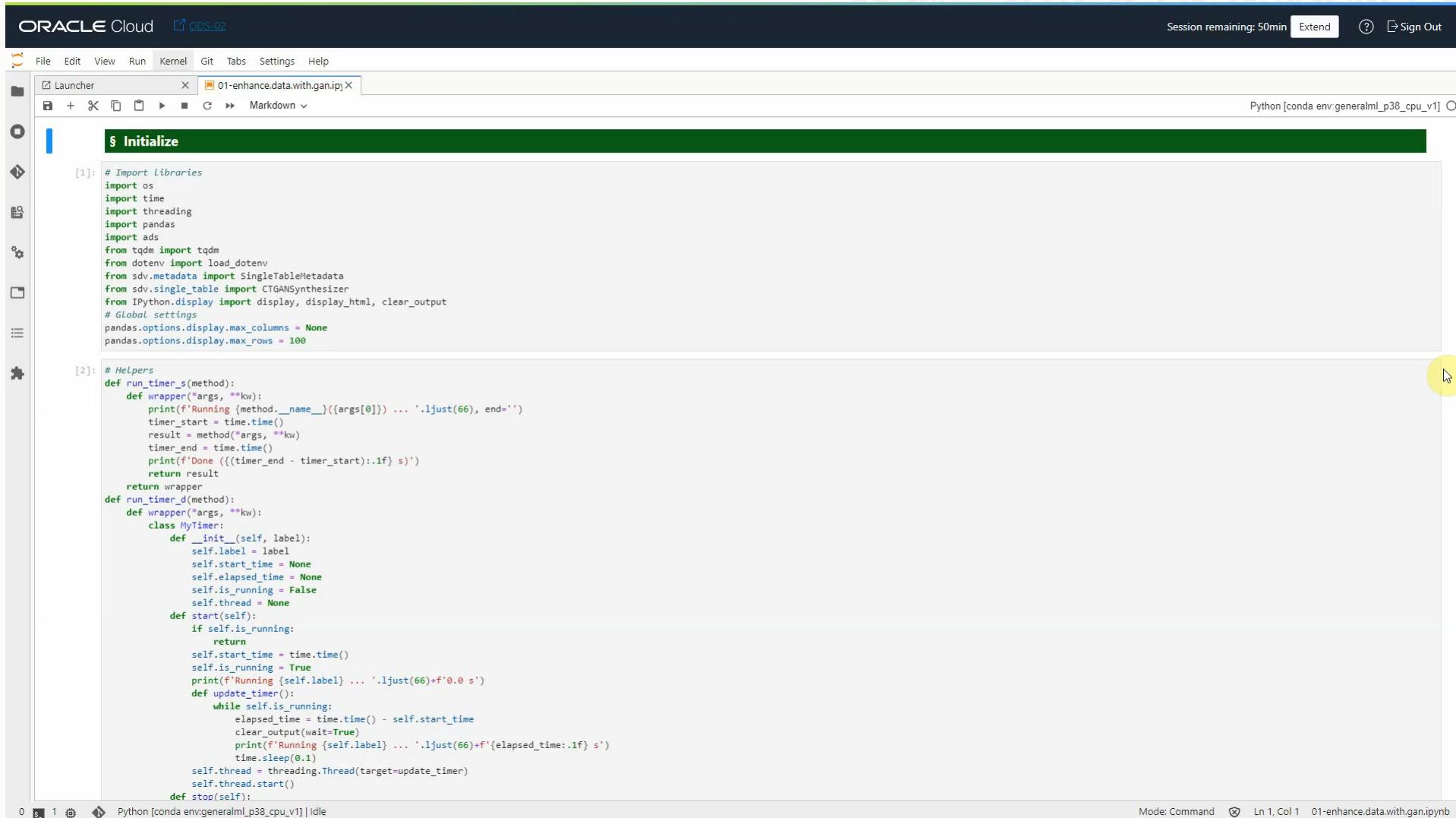
```
[18]: 1 # Explore shape
2 data_shape_1 = get_shape_for_tables(data_1, tables)
3 # Explore content
4 data_content_1 = get_content_for_tables(data_1, tables)
```

S: 100% [██████████] 2/2 [00:00<00:00, 315.86 it/s]
C: 100% [██████████] 2/2 [00:00<00:00, 43.22 it/s]

```
[19]: 1 # Helpers
2 def display_single(df, head):
3     df_styler = df.head(head).style.set_table_attributes("style='display:inline'")
4     df_styler = df_styler.set_properties(subset=['TABLE', 'COLUMN', 'TOPVALUES'], **{'text-align':'left'}).set_table_styles([dict(selector='th', props=[('text-align', 'left')])])
```

Python [conda env:generalml_p38_cpu_v1] | idle Mode: Command ↻ Ln 1, Col 1 01-enhance.data.with.gan.ipynb

Data Enhancement with Generative AI



ORACLE Cloud QDS-02

Session remaining: 50min Extend ? Sign Out

File Edit View Run Kernel Git Tabs Settings Help

Launcher 01-enhance.data.with.gan.ipynb

Python [conda env:generalml_p38_cpu_v1]

§ Initialize

```
[1]: # Import Libraries
import os
import time
import threading
import pandas
import ads
from tqdm import tqdm
from dotenv import load_dotenv
from sdv.metadata import SingleTableMetadata
from sdv.single_table import CTGANsynthesizer
from IPython.display import display, display_html, clear_output
# Global settings
pandas.options.display.max_columns = None
pandas.options.display.max_rows = 100

[2]: # Helpers
def run_timer_s(method):
    def wrapper(*args, **kw):
        print(f'Running {method.__name__}({args[0]}) ... .ljust(66), end="")')
        timer_start = time.time()
        result = method(*args, **kw)
        timer_end = time.time()
        print(f'Done {(timer_end - timer_start):.1f} s')
        return result
    return wrapper
def run_timer_d(method):
    def wrapper(*args, **kw):
        class MyTimer:
            def __init__(self, label):
                self.label = label
                self.start_time = None
                self.elapsed_time = None
                self.is_running = False
                self.thread = None
            def start(self):
                if self.is_running:
                    return
                self.start_time = time.time()
                self.is_running = True
                print(f'Running {self.label} ... .ljust(66)+f'0.0 s')
            def update_timer():
                while self.is_running:
                    elapsed_time = time.time() - self.start_time
                    clear_output(wait=True)
                    print(f'Running {self.label} ... .ljust(66)+f'{elapsed_time:.1f} s')
                    time.sleep(0.1)
                    self.thread = threading.Thread(target=update_timer)
                    self.thread.start()
            def stop(self):
                if self.is_running:
                    self.is_running = False
                    self.thread.join()
        return MyTimer()
    return wrapper
```

0 s 1 ⏪ Python [conda env:generalml_p38_cpu_v1] | idle Mode: Command ⌂ Ln 1, Col 1 01-enhance.data.with.gan.ipynb

Data Exploration



Data Scientist explores the data to have a good understanding before proceeding with more advanced analysis and modeling tasks.

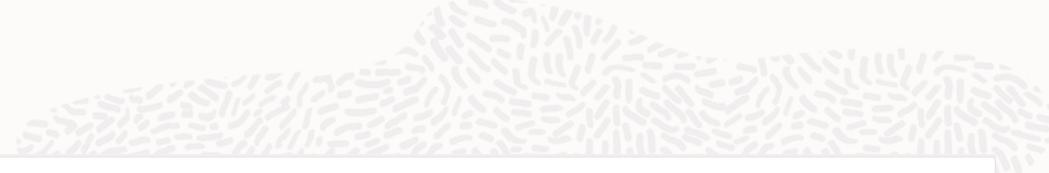
Companies	Transactions
Company ID	Transaction ID
Company Name	Company From ID
...	Company To ID
	Transaction Date
	Transaction Amount
	...

200 companies

10K transactions



Data Exploration



```
1 %python
2 z.show(TF_WIP_SHAPE)
3
```

FINISHED

settings ▾

TABLE	ROWS	COLUMNS
TF_COMPANIES	200	2
TF_TRANSACTIONS	10000	5

```
1 %python
2 z.show(TF_WIP_CONTENT)
3
```

FINISHED

settings ▾

TABLE	COLUMN	VALUES	NULLS	NOTNULLS	DISTINCT	TOPVALUES
TF_COMPANIES	COMPANYID	200	0	200	200	C310023084 : 1 (0.5%), C307110381 : 1 (0.5%), C306592341 : 1 (0.5%), C306634688 : 1 (0.5%), C306714099 : 1 (0.5%),
TF_COMPANIES	COMPANYNAME	200	0	200	200	Company #0079 : 1 (0.5%), Company #0016 : 1 (0.5%), Company #0006 : 1 (0.5%), Company #0007 : 1 (0.5%), Company #0008 : 1 (0.5%),
TF_TRANSACTIONS	TRANSACTIONID	10000	0	10000	10000	T6438000001 : 1 (0.0%), T6438006671 : 1 (0.0%), T6438006664 : 1 (0.0%), T6438006665 : 1 (0.0%), T6438006666 : 1 (0.0%),
TF_TRANSACTIONS	TRANSACTIONDATE	10000	0	10000	31	13.04.2017 : 390 (3.9%), 12.04.2017 : 388 (3.9%), 23.04.2017 : 384 (3.8%), 07.04.2017 : 372 (3.7%), 14.04.2017 : 363 (3.6%),
TF_TRANSACTIONS	COMPANYFROMID	10000	0	10000	200	C315346233 : 143 (1.4%), C309440200 : 138 (1.4%), C307110381 : 124 (1.2%), C309113624 : 116 (1.2%), C306429876 : 105 (1.1%),
TF_TRANSACTIONS	COMPANYTOID	10000	0	10000	200	C313789101 : 104 (1.0%), C313862631 : 100 (1.0%), C315734991 : 98 (1.0%), C310308434 : 97 (1.0%), C315664164 : 96 (1.0%),
TF_TRANSACTIONS	TRANSACTIONAMOUNT	10000	0	10000	6465	49 : 3158 (31.6%), 2948 : 5 (0.1%), 10335 : 4 (0.0%), 882 : 3 (0.0%), 622 : 3 (0.0%),

Data Exploration

The screenshot shows the Oracle Machine Learning Data Explorer interface with two main workspace sections:

- Initialize**: Took 0 secs. Last updated by VALENTIN at September 11 2023, 12:43:16 PM. (outdated)
Code:

```
1 %python
2
3 # check resources
4 def check_compute_performance(label, scale):
5     result = 0
6     for i in range(scale):
7         for j in range(scale):
8             result = result + i * j
9     check_compute_performance('ADW.OML', 10000)
```
- Check data (enhanced)**: Took 1 sec. Last updated by VALENTIN at September 14 2023, 10:15:15 AM. (outdated)
Code:

```
1 %sql
2
3 select OWNER, TABLE_NAME from all_tables where OWNER like 'VALENTIN' order by OWNER, TABLE_NAME
```

OWNER	TABLE_NAME
VALENTIN	ADB_GRAPH_SETTINGS\$
VALENTIN	TF_COMPANIES
VALENTIN	TF_PGE_TRANSACTIONS
VALENTIN	TF_PGV_COMPANIES
VALENTIN	TF_PG_TAXFRAUD_ELEM_TABLE\$
VALENTIN	TF_PG_TAXFRAUD_KEY\$
VALENTIN	TF_PG_TAXFRAUD_LABEL\$
VALENTIN	TF_PG_TAXFRAUD_PROPERTY\$


```
1 %sql
2
3 select count(1) as "CNT" from TF_TRANSACTIONS
```

CNT
10000

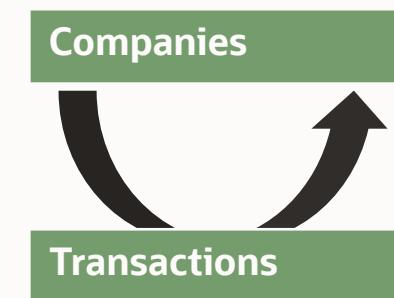
Data Preparation (for Graph)



Data Scientist transforms tabular data into a format suitable for representing companies as nodes and transaction amount as edges in a graph.

Companies
Company ID
Company Name
...

Transactions
Transaction ID
Company From ID
Company To ID
Transaction Date
Transaction Amount
...



Data Preparation (for Graph)

```
1 %python
2
3 # imports
4 import warnings
5 import omi
6 import pandas
7
8 # settings
9 warnings.filterwarnings('ignore')
10 pandas.set_option('display.width', 2000)
```

FINISHED

```
1 %python
2
3 # sync proxy objects
4 TF_COMPANIES = omi.sync(table="TF_COMPANIES")
5 TF_TRANSACTIONS = omi.sync(table="TF_TRANSACTIONS")
6
7 # pull data
8 data = {}
9 data['TF_COMPANIES'] = TF_COMPANIES.pull()
10 data['TF_TRANSACTIONS'] = TF_TRANSACTIONS.pull()
```

FINISHED

```
1 %python
2
3 # init
4 data_pg = {}
5
6 # transform data 'COMPANIES'
7 data_pg['TF_PGV_COMPANIES'] = data['TF_COMPANIES']
8
9 # transform data 'TRANSACTIONS'
10 data_pg['TF_PGE_TRANSACTIONS'] = data['TF_TRANSACTIONS']
11
12 # check
13 print(f'COMPANIES data now has {data_pg["TF_PGV_COMPANIES"].shape[0]} rows and {data_pg["TF_PGV_COMPANIES"].shape[1]} columns.')
14 print(f'TRANSACTIONS data now has {data_pg["TF_PGE_TRANSACTIONS"].shape[0]} rows and {data_pg["TF_PGE_TRANSACTIONS"].shape[1]} columns.')
15
```

COMPANIES data now has 200 rows and 2 columns.
TRANSACTIONS data now has 10000 rows and 5 columns.

```
1 %python
2
3 # init
4 data_s = data_pg
5
6 # push data
7 try:
8     omi.drop(table='TF_PGV_COMPANIES')
9     omi.drop(table='TF_PGE_TRANSACTIONS')
10 except:
11     pass
12 TF_PGV_COMPANIES = omi.create(data_s['TF_PGV_COMPANIES'], table='TF_PGV_COMPANIES',
13                                dbtypes={'COMPANYID':'VARCHAR2(100)', 'COMPANYNAME':'VARCHAR2(100)'})
14 TF_PGE_TRANSACTIONS = omi.create(data_s['TF_PGE_TRANSACTIONS'], table='TF_PGE_TRANSACTIONS',
15                                dbtypes={'TRANSACTIONID':'VARCHAR2(100)', 'TRANSACTIONDATE':'VARCHAR2(100)', 'COMPANYFROMID':'VARCHAR2(100)', 'COMPANYTOID':'VARCHAR2(100)', 'TRANSACTIONAMOUNT':'NUMBER(10)'})
```

FINISHED

```
1 %script
2
3 alter table TF_PGE_TRANSACTIONS drop constraint FK_TF_PGE_TRANSACTIONS_FROM;
4 alter table TF_PGE_TRANSACTIONS drop constraint FK_TF_PGE_TRANSACTIONS_TO;
5 alter table TF_PGE_TRANSACTIONS drop constraint PK_TF_PGE_TRANSACTIONS;
6 alter table TF_PGV_COMPANIES drop constraint PK_TF_PGV_COMPANIES
```

FINISHED

```
1 %script
2
3 alter table TF_PGV_COMPANIES add constraint PK_TF_PGV_COMPANIES PRIMARY KEY (COMPANYID);
4 alter table TF_PGE_TRANSACTIONS add constraint PK_TF_PGE_TRANSACTIONS PRIMARY KEY (TRANSACTIONID);
5 alter table TF_PGE_TRANSACTIONS add constraint FK_TF_PGE_TRANSACTIONS_FROM FOREIGN KEY (COMPANYFROMID) references TF_PGV_COMPANIES(COMPANYID);
6 alter table TF_PGE_TRANSACTIONS add constraint FK_TF_PGE_TRANSACTIONS_TO FOREIGN KEY (COMPANYTOID) references TF_PGV_COMPANIES(COMPANYID)
```

FINISHED



Data Preparation (for Graph)

The screenshot shows the Oracle Machine Learning interface with a project named "VALENTIN Project" and workspace "VALENTIN". A large watermark of a brain is visible on the right side.

The main area displays three data preparation steps:

- 03-prepare.data.for.graph**:
 - § Initialize**: Took 0 secs. Last updated by VALENTIN at September 11 2023, 12:43:16 PM. (outdated)
%python
check resources
def check_compute_performance(label, scale):
 result = 0
 for i in range(scale):
 for j in range(scale):
 result = result + i * j
 check_compute_performance('ADW.OML', 10000)
Took 35 secs. Last updated by VALENTIN at September 26 2023, 9:10:21 AM.
 - FINISHED**
- § Transform data (for graph)**: Took 1 sec. Last updated by VALENTIN at September 14 2023, 10:18:55 AM. (outdated)
 - FINISHED**
 - FINISHED**
- FINISHED**

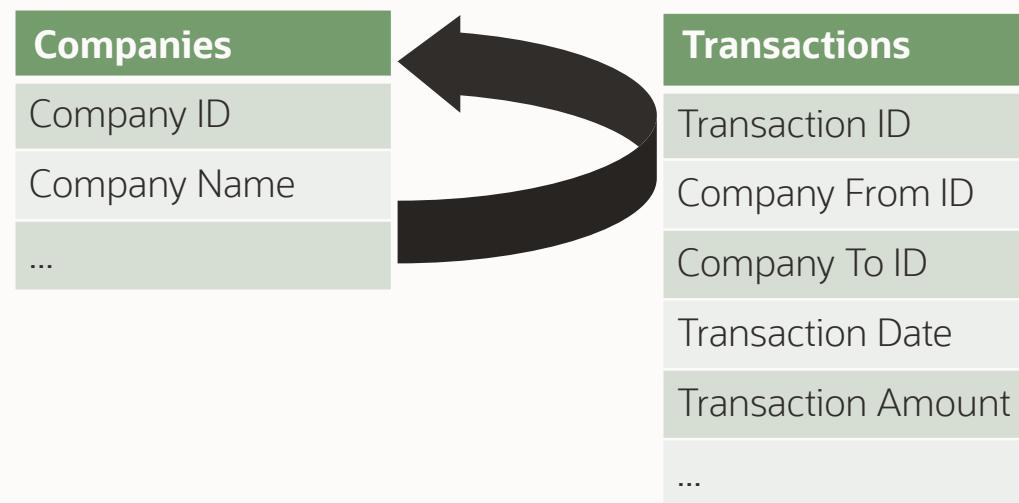
At the bottom, there is a third section of code:

```
1 %python
2
3 # init
4 data_pg = {}
5
6 # transform data 'COMPANIES'
7 data_pg['TF_PGV_COMPANIES'] = data['TF_COMPANIES']
8
9 # transform data 'TRANSACTIONS'
10 data_pg['TF_PGE_TRANSACTIONS'] = data['TF_TRANSACTIONS']
11
12 # check
13 print(f'COMPANIES data now has {data_pg["TF_PGV_COMPANIES"].shape[0]} rows and {data_pg["TF_PGV_COMPANIES"].shape[1]} columns.')
14 print(f'TRANSACTIONS data now has {data_pg["TF_PGE_TRANSACTIONS"].shape[0]} rows and {data_pg["TF_PGE_TRANSACTIONS"].shape[1]} columns.'
```

Graph Model



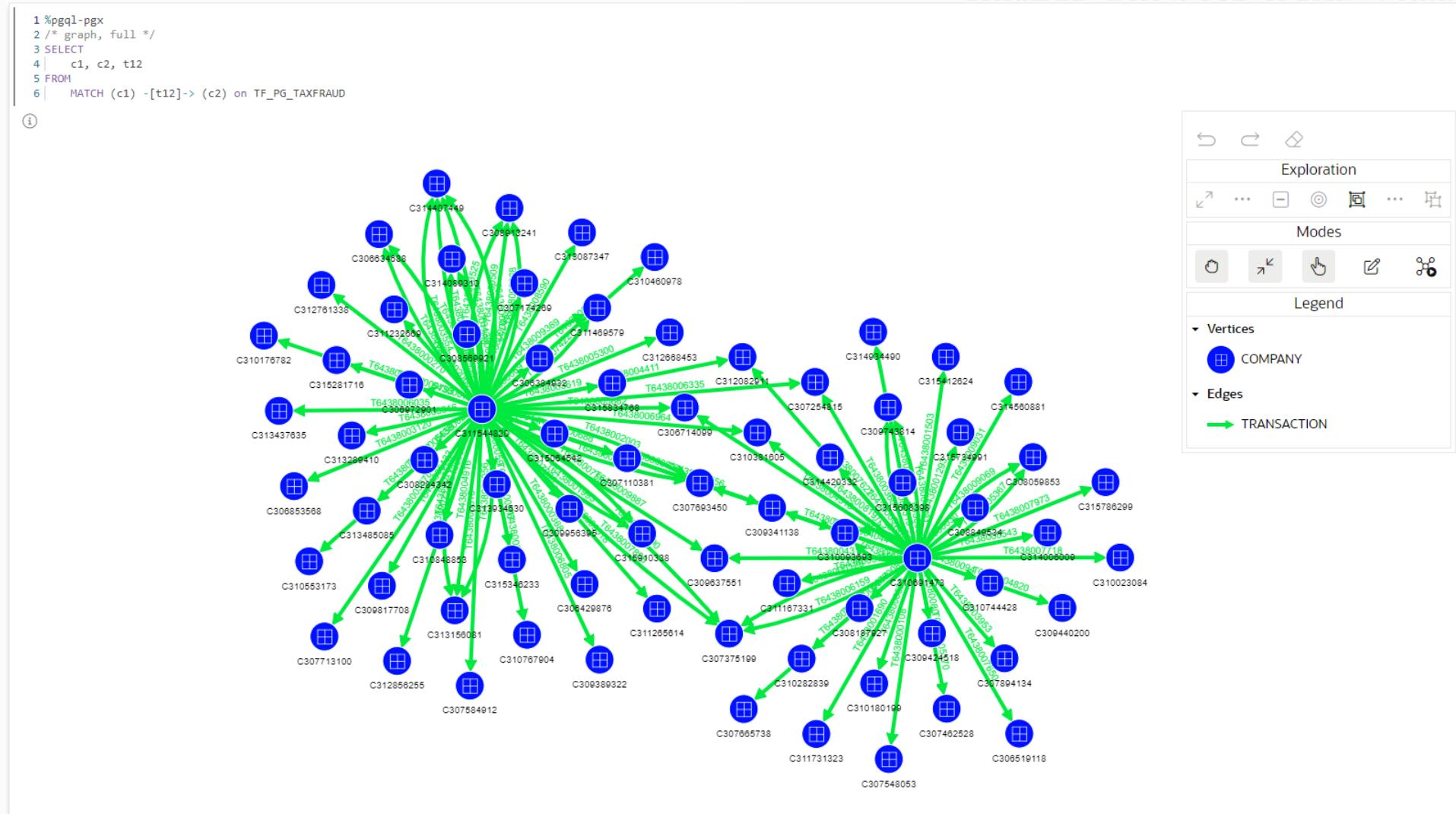
Data Scientist builds a Graph model to investigate cyclic money transfers and companies who are at the center of these transactions.



200 nodes

10K edges

Graph Model



Graph Model

The screenshot shows the Oracle Graph Studio interface. At the top, there's a navigation bar with a user icon and the name "VALENTIN". On the left, a sidebar has icons for Home, Create, Edit, and Delete. The main area has a title "Get Started" with three sections: "Graphs", "Notebooks", and "Jobs". Each section has an icon, a title, and a brief description. Below these are two large sections: "Graphs" and "Notebooks", each with a "View All" button.

Graph Studio

VALENTIN

Get Started

Graphs
Create, query, and analyze graphs from tables in your Autonomous Database.
Select Tables —● Create Graph —● Analyze

Notebooks
Create notebooks to run code and work interactively with graphs.

Jobs
Jobs running in Graph Studio.

Welcome to Oracle Graph

Simplify Graph Analytics with Autonomous Database

Learn More

Documentation, Blog, Tutorials

Graphs

TF_PG_TAXFRAUD
Graph | VALENTIN
Updated 8 days ago

Notebooks

View All

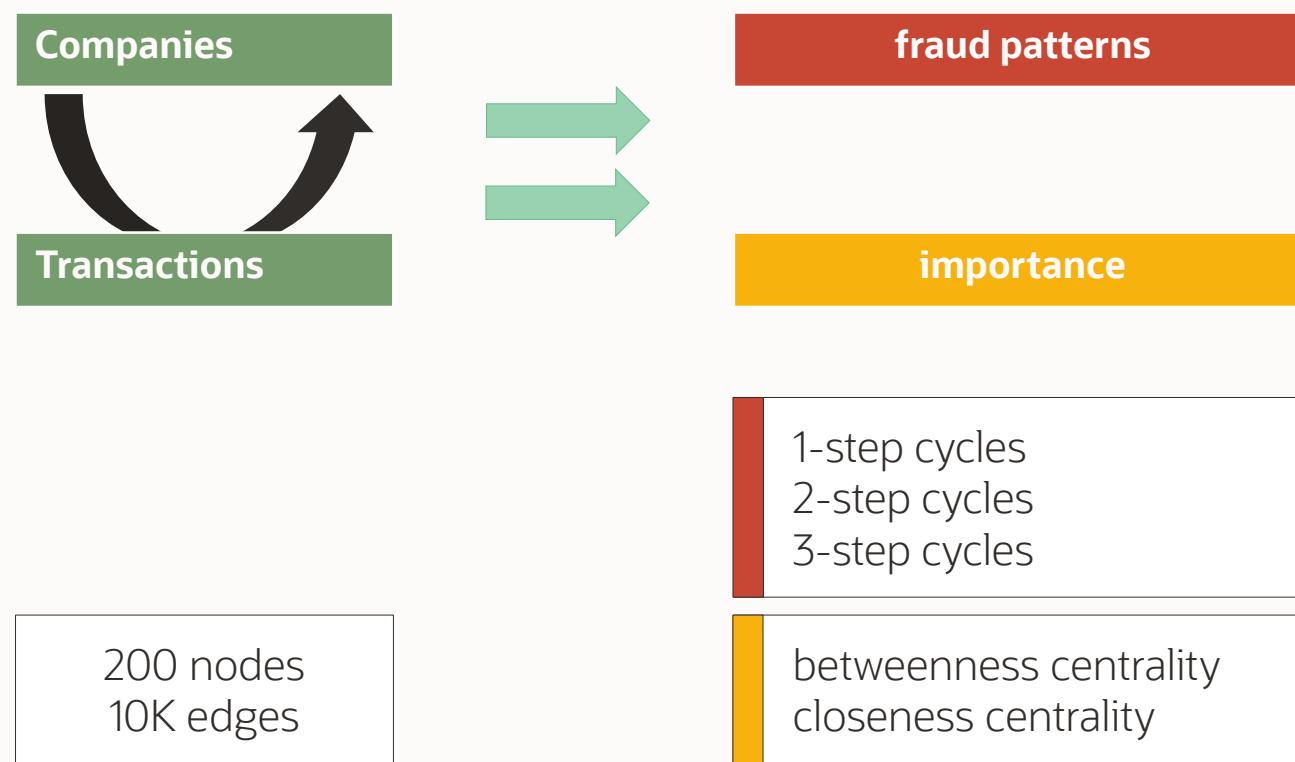
View All

Copyright © 2025 Oracle and/or its affiliates.

Graph Metrics



Data Scientist identifies 1-, 2- and 3-step cycles of transactions as well as calculates importance scores of the companies.



Graph Metrics

ID	NAME	SCORE
C309440200	Company #0064	59
C313862631	Company #0154	45
C312947258	Company #0135	42
C306429876	Company #0002	41
C308753692	Company #0048	36

ID	NAME	SCORE
C309440200	Company #0064	2733
C313862631	Company #0154	2088
C306429876	Company #0002	1721
C312947258	Company #0135	1665
C313934530	Company #0155	1582

ID	NAME	SCORE
C309440200	Company #0064	134808
C313862631	Company #0154	103939
C306429876	Company #0002	90206
C312947258	Company #0135	82454
C313934530	Company #0155	81816

ID	NAME	SCORE
C309440200	Company #0064	712.1449272244015
C313862631	Company #0154	537.4907683695352
C306429876	Company #0002	444.29669081790325
C315346233	Company #0186	423.10032538396234
C312947258	Company #0135	401.4715513258675

ID	NAME	SCORE
C309440200	Company #0064	0.0038461538461538464
C313862631	Company #0154	0.0035587188612099642
C306429876	Company #0002	0.0035211267605633804
C315346233	Company #0186	0.0034965034965034965
C307110381	Company #0016	0.003484320557491289

Graph Metrics

The screenshot shows the Oracle Graph Studio interface with two code snippets displayed in separate panes.

Initialization:

```
1 %java-pgx
2 /* graph */
3 PgxGraph pg = session.getGraph("TF_PG_TAXFRAUD");
4 pg
```

PgxGraph[name=TF_PG_TAXFRAUD,N=200,E=10000,created=1695106205781]

Extract metrics (graph queries):

```
1 %pgql-pgx
2 /* 1-step cycle, (A)->(B)->(A) */
3 SELECT
4     a.COMPANYID as "ID",
5     a.COMPANYNAME as "NAME",
6     count(*) as "SCORE"
7 FROM
8     MATCH (a) -[t1]-> (b) on TF_PG_TAXFRAUD,
9     MATCH (b) -[t2]-> (a) on TF_PG_TAXFRAUD
10 GROUP BY a
11 ORDER BY count(*) desc
```

ID	NAME	SCORE
C309440200	Company #0064	59

```
1 %java-pgx
2 /* 1-step cycle, (A)->(B)->(A) */
3 var query = "SELECT" +
4     " a.COMPANYID as \"ID\", " +
5     " a.COMPANYNAME as \"NAME\", " +
6     " count(*) as \"SCORE\" " +
7     " FROM" +
8     "     MATCH (a) -[t1]-> (b) on TF_PG_TAXFRAUD," +
9     "     MATCH (b) -[t2]-> (a) on TF_PG_TAXFRAUD" +
10    " GROUP BY a"
11 var result = pg.queryPgql(query)
12 var table = "TF_WIP_PG_CYCLE1"
13 result.toFrame().write().db().tablename(table).overwrite(true).store()
```

Successful run: No result returned.

Copyright © 2023 Oracle and/or its affiliates.

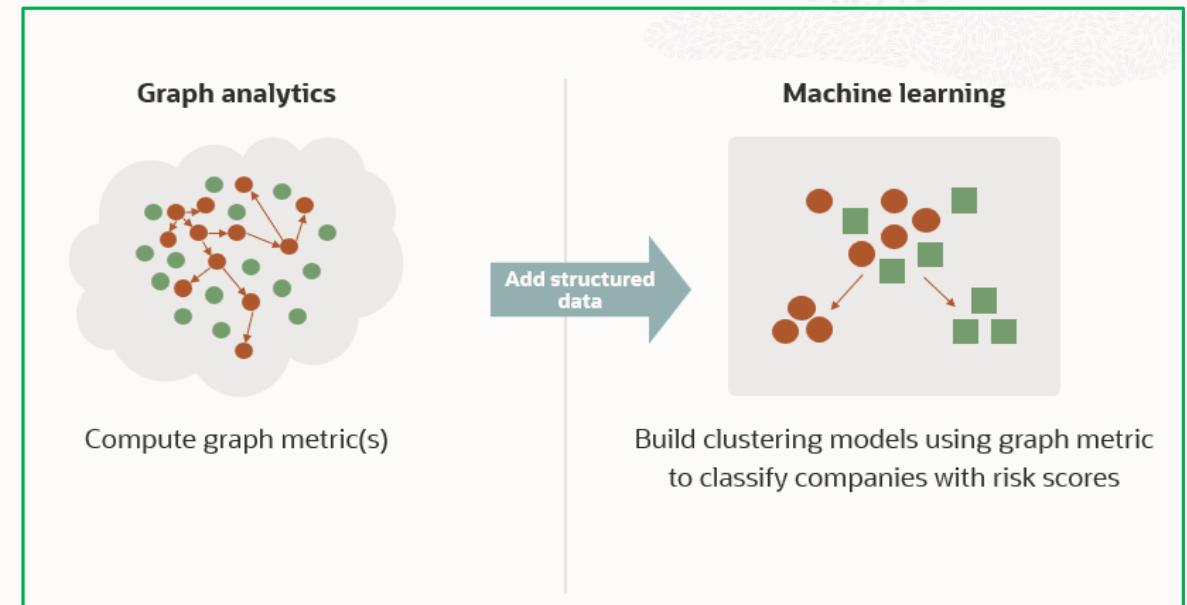
Detecting Companies with High Fraud Risk



Fraud Investigator would like to understand the list of companies with high fraud potential.



Data Scientist uses Graph metrics to segment companies to provide higher risk companies to fraud investigator.



Data Preparation (for Machine Learning)



Data Scientist prepares data for segmenting companies in terms of fraud risks.

Companies



Companies

1-step cycles score

2-step cycles score

3-step cycles score

betweenness centrality score

closeness centrality score

1-step cycles
2-step cycles
3-step cycles

betweenness centrality
closeness centrality

Data Preparation (for Machine Learning)



```
1 %python
2
3 # imports
4 import warnings
5 import oml
6 import pandas
7
8 # settings
9 warnings.filterwarnings('ignore')
10 pandas.set_option('display.width', 2000)
```

```
1 %python
2
3 # sync proxy objects
4 TF_WIP_PG_CYCLE1 = om1.sync(table="TF_WIP_PG_CYCLE1")
5 TF_WIP_PG_CYCLE2 = om1.sync(table="TF_WIP_PG_CYCLE2")
6 TF_WIP_PG_CYCLE3 = om1.sync(table="TF_WIP_PG_CYCLE3")
7 TF_WIP_PG_BCENTRAL = om1.sync(table="TF_WIP_PG_BCENTRAL")
8 TF_WIP_PG_CCENTRAL = om1.sync(table="TF_WIP_PG_CCENTRAL")
9
10 # pull data
11 data = {}
12 data['TF_WIP_PG_CYCLE1'] = TF_WIP_PG_CYCLE1.pull()
13 data['TF_WIP_PG_CYCLE2'] = TF_WIP_PG_CYCLE2.pull()
14 data['TF_WIP_PG_CYCLE3'] = TF_WIP_PG_CYCLE3.pull()
15 data['TF_WIP_PG_BCENTRAL'] = TF_WIP_PG_BCENTRAL.pull()
16 data['TF_WIP_PG_CCENTRAL'] = TF_WIP_PG_CCENTRAL.pull()
```

```
1 %python
2
3 # transform data 'METRICS'
4 data['TF_WIP_PG_METRICS'] = pandas.merge(data['TF_WIP_PG_CYCLE1'], data['TF_WIP_PG_CYCLE2'], left_on=['ID'], right_on=['ID'], how='inner')
5 data['TF_WIP_PG_METRICS'] = data['TF_WIP_PG_METRICS'].drop(columns=['NAME_y'])
6 data['TF_WIP_PG_METRICS'] = data['TF_WIP_PG_METRICS'].rename(columns={'NAME_x': 'NAME', 'SCORE_x': 'SCORE_CYC1', 'SCORE_y': 'SCORE_CYC2'})
7 data['TF_WIP_PG_METRICS'] = pandas.merge(data['TF_WIP_PG_METRICS'], data['TF_WIP_PG_CYCLE3'], left_on=['ID'], right_on=['ID'], how='inner')
8 data['TF_WIP_PG_METRICS'] = data['TF_WIP_PG_METRICS'].drop(columns=['NAME_y'])
9 data['TF_WIP_PG_METRICS'] = data['TF_WIP_PG_METRICS'].rename(columns={'NAME_x': 'NAME', 'SCORE': 'SCORE_CYC3'})
10 data['TF_WIP_PG_METRICS'] = pandas.merge(data['TF_WIP_PG_METRICS'], data['TF_WIP_PG_BCENTRAL'], left_on=['ID'], right_on=['ID'], how='inner')
11 data['TF_WIP_PG_METRICS'] = data['TF_WIP_PG_METRICS'].drop(columns=['NAME_y'])
12 data['TF_WIP_PG_METRICS'] = data['TF_WIP_PG_METRICS'].rename(columns={'NAME_x': 'NAME', 'SCORE': 'SCORE_BCEN'})
13 data['TF_WIP_PG_METRICS'] = pandas.merge(data['TF_WIP_PG_METRICS'], data['TF_WIP_PG_CCENTRAL'], left_on=['ID'], right_on=['ID'], how='inner')
14 data['TF_WIP_PG_METRICS'] = data['TF_WIP_PG_METRICS'].drop(columns=['NAME_y'])
15 data['TF_WIP_PG_METRICS'] = data['TF_WIP_PG_METRICS'].rename(columns={'ID': 'COMPANYID', 'NAME_x': 'COMPANYNAME', 'SCORE': 'SCORE_CCEN'})
16 for col in ['SCORE_CYC1', 'SCORE_CYC2', 'SCORE_CYC3', 'SCORE_BCEN', 'SCORE_CCEN']:
17     data['TF_WIP_PG_METRICS'][col] = round(100 * (data['TF_WIP_PG_METRICS'][col] - data['TF_WIP_PG_METRICS'][col].min()) / (data['TF_WIP_PG_METRICS'][col].max() - data['TF_WIP_PG_METRICS'][col].min()), 2)
18
19 # check
20 z.show(data['TF_WIP_PG_METRICS'])
```

COMPANYID	COMPANYNAME	SCORE_CYC1	SCORE_CYC2	SCORE_CYC3	SCORE_BCEN	SCORE_CCEN
C308732444	Company #0047	3.45	7.82	7.15	8.48	31.14
C309830621	Company #0074	5.17	6.41	5.33	4.49	15.3
C307940687	Company #0034	5.17	11.63	11.72	12.45	27.97
C310691473	Company #0091	6.9	6.04	6.19	3.7	16.01
C309204505	Company #0056	8.62	16.3	17.65	11.13	28.75
C314465049	Company #0171	10.34	11.04	10.89	10.5	33.57
C308125507	Company #0037	12.07	14.97	16.44	17.81	33.57
C309341138	Company #0059	12.07	15.56	15.91	17.88	35.22

Data Preparation (for Machine Learning)

The screenshot shows the Oracle Machine Learning interface with two parallel data preparation jobs running in separate windows.

Job 1: Initialize

Code:

```
1 %python
2
3 # check resources
4 def check_compute_performance(label, scale):
5     result = 0
6     for i in range(scale):
7         for j in range(scale):
8             result = result + i * j
9     check_compute_performance('ADW.OML', 10000)
```

Output:

```
Took 0 secs. Last updated by VALENTIN at September 11 2023, 12:43:16 PM. (outdated)
```

Job 2: Transform data (from graph)

Code:

```
1 %script
2
3 create table TF_WIP_PG_BCENTRAL_TMP as
4     (select ID, max(NAME) as "NAME", max(SCORE) as "SCORE" from TF_WIP_PG_BCENTRAL group by ID);
5 drop table TF_WIP_PG_BCENTRAL;
6 create table TF_WIP_PG_BCENTRAL as (select * from TF_WIP_PG_BCENTRAL_TMP);
7 drop table TF_WIP_PG_BCENTRAL_TMP
```

Output:

```
Took 1 sec. Last updated by VALENTIN at September 19 2023, 8:52:27 AM. (outdated)
```

Job 3: Transform data (from graph)

Code:

```
1 %script
2
3 create table TF_WIP_PG_CCENTRAL_TMP as
4     (select ID, max(NAME) as "NAME", max(SCORE) as "SCORE" from TF_WIP_PG_CCENTRAL group by ID);
5 drop table TF_WIP_PG_CCENTRAL;
6 create table TF_WIP_PG_CCENTRAL as (select * from TF_WIP_PG_CCENTRAL_TMP);
7 drop table TF_WIP_PG_CCENTRAL_TMP
```

Output:

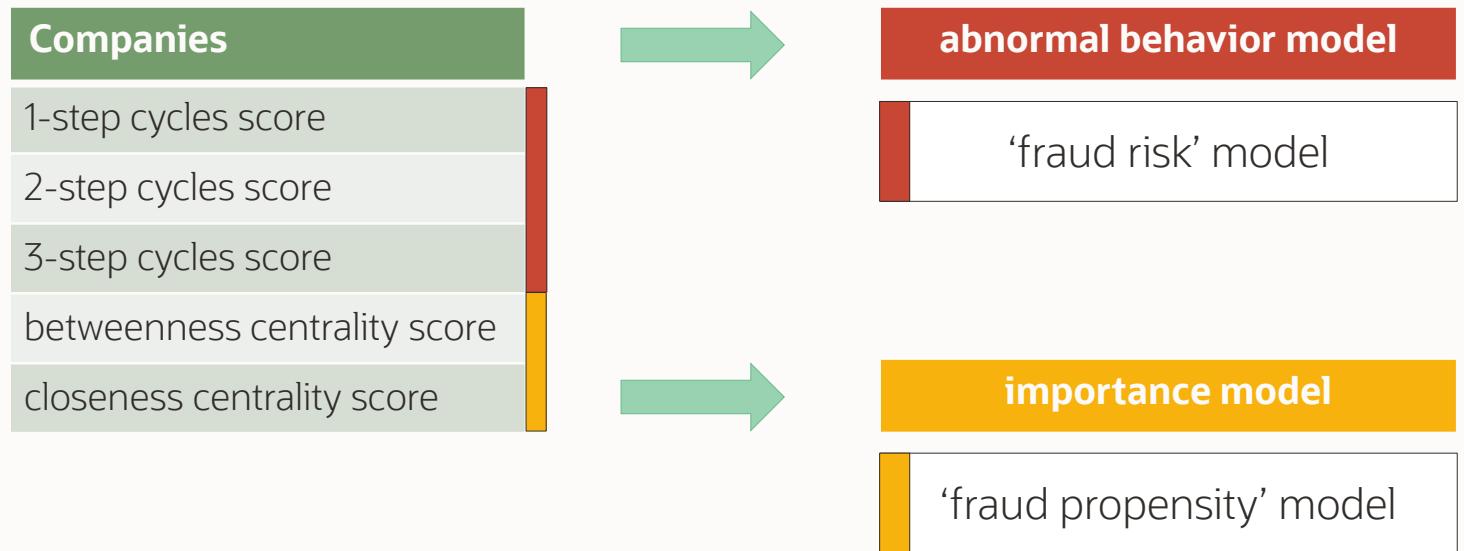
```
Took 1 sec. Last updated by VALENTIN at September 19 2023, 8:52:27 AM. (outdated)
```

The interface includes a top navigation bar with 'VALENTIN Project' and 'VALENTIN Workspace' tabs, and a status indicator 'Connected'.

Machine Learning Model



Data Scientist builds two machine learning models to detect fraud risk and propensity for fraud of the companies.



Machine Learning Model

FINISHED ▶ ✎ ⚙️

```
1 %python
2
3 # prepare data
4 TF_WIP_CL_CYCLES_IN = om1.push(data['TF_WIP_PG_METRICS'][[['COMPANYID','SCORE_CYC1','SCORE_CYC2','SCORE_CYC3']])
5 TF_WIP_CL_CENTRAL_IN = om1.push(data['TF_WIP_PG_METRICS'][[['COMPANYID','SCORE_BCEN','SCORE_CCEN']])
6
7 # build models
8 try:
9     om1.drop(model="TF_WIP_CL_CYCLES")
10    om1.drop(model="TF_WIP_CL_CENTRAL")
11 except:
12     pass
13 setting = {'KMNS_ITERATIONS':100, 'KMNS_RANDOM_SEED':6}
14 TF_WIP_CL_CYCLES = om1.km(n_clusters=5, **setting).fit(TF_WIP_CL_CYCLES_IN, case_id='COMPANYID', model_name="TF_WIP_CL_CYCLES")
15 TF_WIP_CL_CENTRAL = om1.km(n_clusters=5, **setting).fit(TF_WIP_CL_CENTRAL_IN, case_id='COMPANYID', model_name="TF_WIP_CL_CENTRAL")
```

Model Name: TF_WIP_CL_CYCLES
Model Owner: VALENTIN
Algorithm Name: K-Means
Mining Function: CLUSTERING
Settings:

	setting name	setting value
0	ALGO_NAME	ALGO_KMEANS
1	CLUS_NUM_CLUSTERS	5
2	KMNS_CONV_TOLERANCE	.001
3	KMNS_DETAILS	KMNS_DETAILS_HIERARCHY
4	KMNS_DISTANCE	KMNS_EUCLIDEAN
5	KMNS_ITERATIONS	100
6	KMNS_HTML_DCT_ATTD_SUPPORT	1

Model Name: TF_WIP_CL_CENTRAL
Model Owner: VALENTIN
Algorithm Name: K-Means
Mining Function: CLUSTERING
Settings:

	setting name	setting value
0	ALGO_NAME	ALGO_KMEANS
1	CLUS_NUM_CLUSTERS	5
2	KMNS_CONV_TOLERANCE	.001
3	KMNS_DETAILS	KMNS_DETAILS_HIERARCHY
4	KMNS_DISTANCE	KMNS_EUCLIDEAN
5	KMNS_ITERATIONS	100
6	KMNS_HTML_DCT_ATTD_SUPPORT	1

COMPANYID	SCORE_CYC1	SCORE_CYC2	SCORE_CYC3	SCORE_BCEN	SCORE_CCEN	CLUSTER_CYC	CLUSTER_CEN
C315592736	0	0	0	0	0	Cluster #1	Cluster #1
C310769904	5	2	2	2	7	Cluster #1	Cluster #1
C312022089	2	3	2	2	11	Cluster #1	Cluster #1
C314201266	2	3	2	2	13	Cluster #1	Cluster #1
C311348313	2	4	5	6	20	Cluster #1	Cluster #1
C309639877	3	4	4	3	11	Cluster #1	Cluster #1
C315509232	3	4	4	4	17	Cluster #1	Cluster #1
C312992057	3	5	5	4	15	Cluster #1	Cluster #1
4							

Machine Learning Model

The screenshot shows the Oracle Machine Learning interface with a workflow titled "07-create.fraud.risk.models". The workflow consists of two main sections: "Initialize" and "Create models (fraud risk)".

Initialize:

```
1 %python
2
3 # check resources
4 def check_compute_performance(label, scale):
5     result = 0
6     for i in range(scale):
7         for j in range(scale):
8             result = result + i * j
9 check_compute_performance('ADW.OHIL', 10000)
```

Took 0 secs. Last updated by VALENTIN at September 11 2023, 12:43:16 PM. (outdated)

Create models (fraud risk):

```
1 %python
2
3 # imports
4 import warnings
5 import om1
6 import pandas
7
8 # settings
9 warnings.filterwarnings('ignore')
10 pandas.set_option('display.width', 2000)
```

Took 1 sec. Last updated by VALENTIN at September 19 2023, 10:29:10 AM. (outdated)1 %python
2
3 # sync proxy objects
4 TF_WIP_PG_METRICS = om1.sync(table="TF_WIP_PG_METRICS")
5
6 # pull data
7 data = {}
8 data['TF_WIP_PG_METRICS'] = TF_WIP_PG_METRICS.pull()

Took 0 secs. Last updated by VALENTIN at September 19 2023, 10:29:58 AM. (outdated)

```
1 %python
2
3 # prepare data
4 TF_WIP_CL_CYCLES_IN = om1.push(data['TF_WIP_PG_METRICS'][['COMPANYID','SCORE_CYC1','SCORE_CYC2','SCORE_CYC3']])
5 TF_WIP_CL_CENTRAL_IN = om1.push(data['TF_WIP_PG_METRICS'][['COMPANYID','SCORE_BCE1','SCORE_CCE1']])
6
7 # build models
8 try:
9     om1.drop(model="TF_WIP_CL_CYCLES")
10    om1.drop(model="TF_WIP_CL_CENTRAL")
11 except:
12     pass
13 setting = {'KMNS_ITERATIONS':100, 'KMNS_RANDOM_SEED':6}
14 TF_WIP_CL_CYCLES = om1.kmns_train(**setting, data=TF_WIP_CL_CYCLES_IN, model='TF_WIP_CL_CYCLES')
```

Took 0 secs. Last updated by VALENTIN at September 19 2023, 10:29:56 AM. (outdated)

The interface includes a header with "VALENTIN Project VALENTIN Workspace" and a status bar indicating "Connected". The bottom right corner features a red square with a white "O" icon.

Business Insights



Data Scientist shares tax fraud segments with **Business Analyst**.



Business Analyst creates dashboards for **Fraud Investigator**.

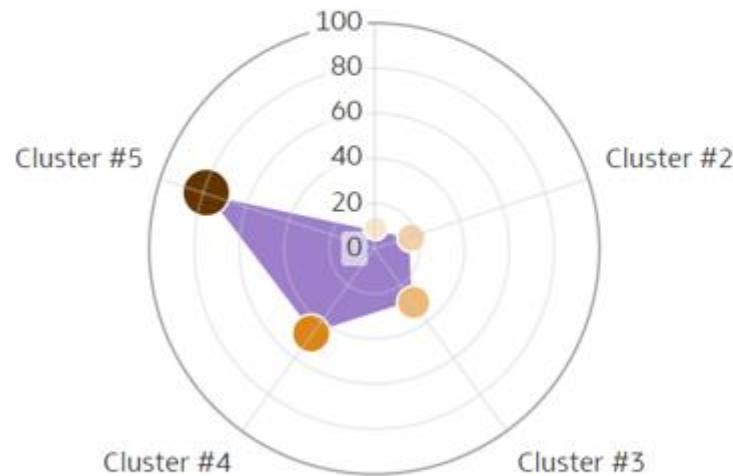


Fraud Investigator accesses all high-risk companies at the click of a button.

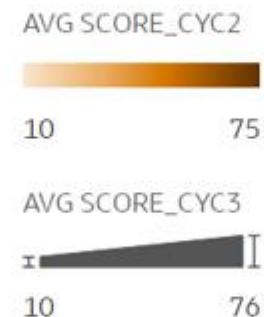
Business Insights



CYCLIC TRANSACTIONS



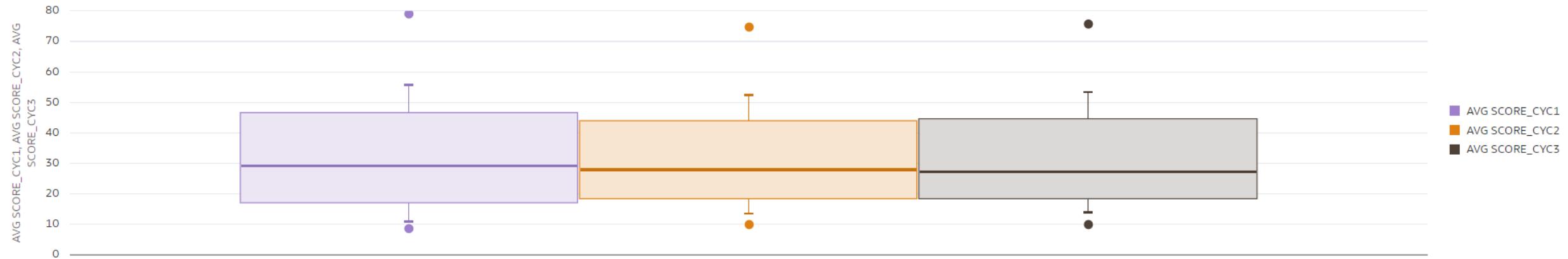
CENTRAL COMPANIES



Business Insights



EXTREME SCORES



HIGH RISK COMPANIES

CLUSTER_CYC: Cluster #5 CLUSTER_CEN: Cluster #5

COMPANYID	CLUSTER_CYC	CLUSTER_CEN
C306429876	Cluster #5	Cluster #5
C309440200	Cluster #5	Cluster #5
C312947258	Cluster #5	Cluster #5
C313862631	Cluster #5	Cluster #5

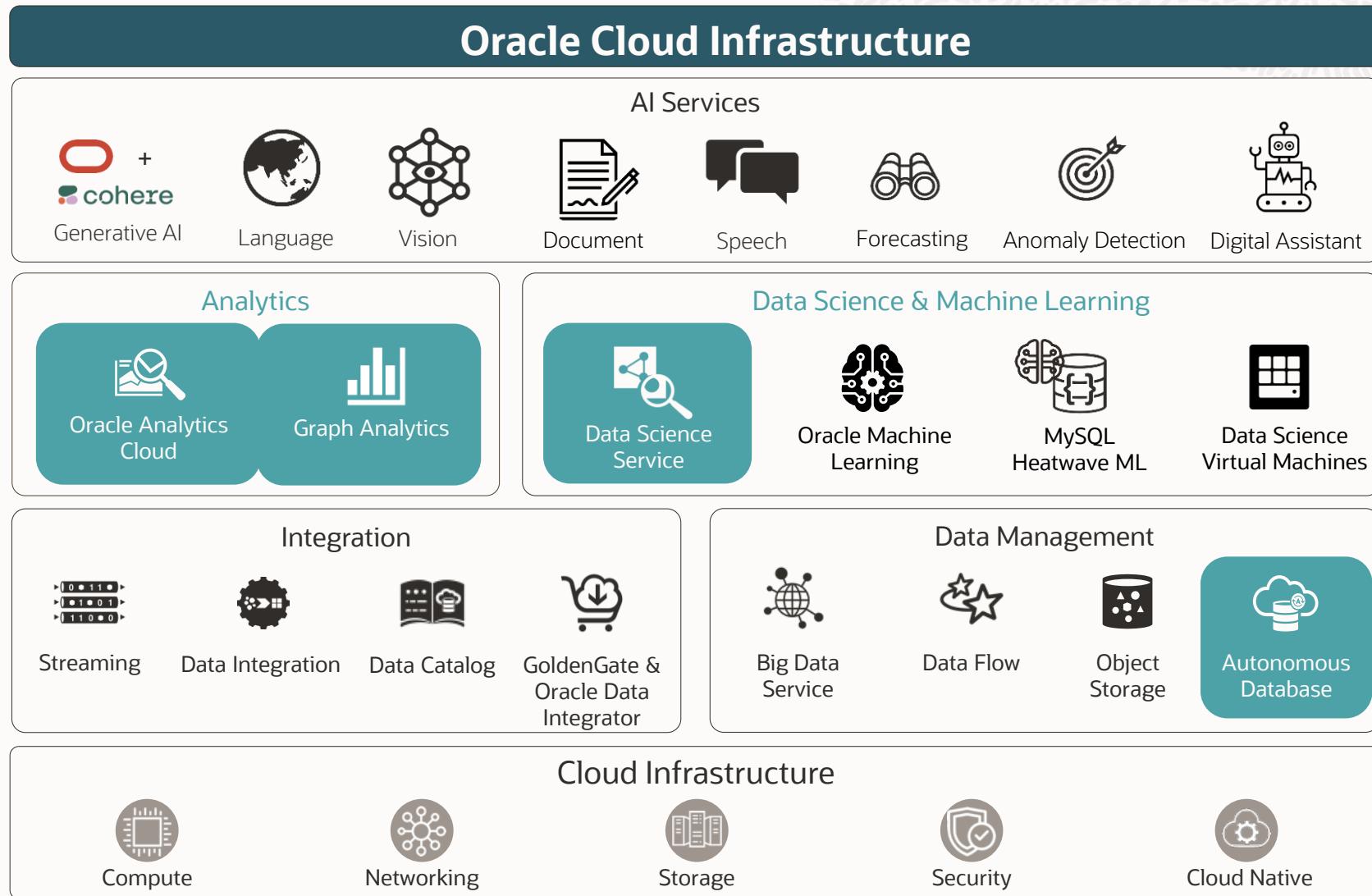
Business Insights

The screenshot shows a web browser window for Oracle Analytics Cloud. The title bar reads "TAX FRAUD". The address bar shows the URL: <https://franceoac-frqap2zhtzbe-fr.analytics.ocp.oraclecloud.com/ui/dv/?pageid=visualAnalyzer&reportmode=full&re...>. The page content displays a table titled "HIGH FRAUD RISK". The table has columns: COMPANYID, CLUSTER_CYC, CLUSTER_CEN, SCORE_CYC1, SCORE_CYC2, SCORE_CYC3, SCORE_BCEN, and SCORE_CCEN. A yellow circle highlights the row for COMPANYID C306558208, which is part of Cluster #4. The table contains 15 rows of data.

COMPANYID	CLUSTER_CYC	CLUSTER_CEN	SCORE_CYC1	SCORE_CYC2	SCORE_CYC3	SCORE_BCEN	SCORE_CCEN
C306429876	Cluster #5	Cluster #5	69	63	66	62	72
C309440200	Cluster #5	Cluster #5	100	100	100	100	100
C312947258	Cluster #5	Cluster #5	71	60	60	56	64
C313862631	Cluster #5	Cluster #5	76	76	77	75	75
C306558208	Cluster #4	Cluster #5	41	49	48	41	63
C306634688	Cluster #4	Cluster #5	53	54	55	47	58
C306778540	Cluster #4	Cluster #5	53	42	45	40	58
C307032812	Cluster #4	Cluster #5	40	40	42	50	64
C307548053	Cluster #4	Cluster #5	53	46	49	40	56
C307894134	Cluster #4	Cluster #5	57	51	53	48	56
C308059853	Cluster #4	Cluster #5	52	47	44	43	60
C308753692	Cluster #4	Cluster #5	60	53	53	48	54
C308849534	Cluster #4	Cluster #5	45	40	40	41	59
C309113624	Cluster #4	Cluster #5	45	47	44	49	65
C311250710	Cluster #4	Cluster #4	70	41	41	70	51

Behind the Scenes

Behind the Scenes



Autonomous Database



Self-Driving

- Scale-out database with fault-tolerance and DR
- Runs on enterprise-proven Exadata platform
- Full compatibility with existing enterprise databases



Self-Securing

- Automatically applies security updates online
- Secure configuration with full database encryption
- Sensitive data hidden from Oracle or customer admins



Self-Repairing

- Recovers automatically from any failure
- 99.995% uptime including maintenance
- Elastically scales compute or storage as needed

Oracle Graph

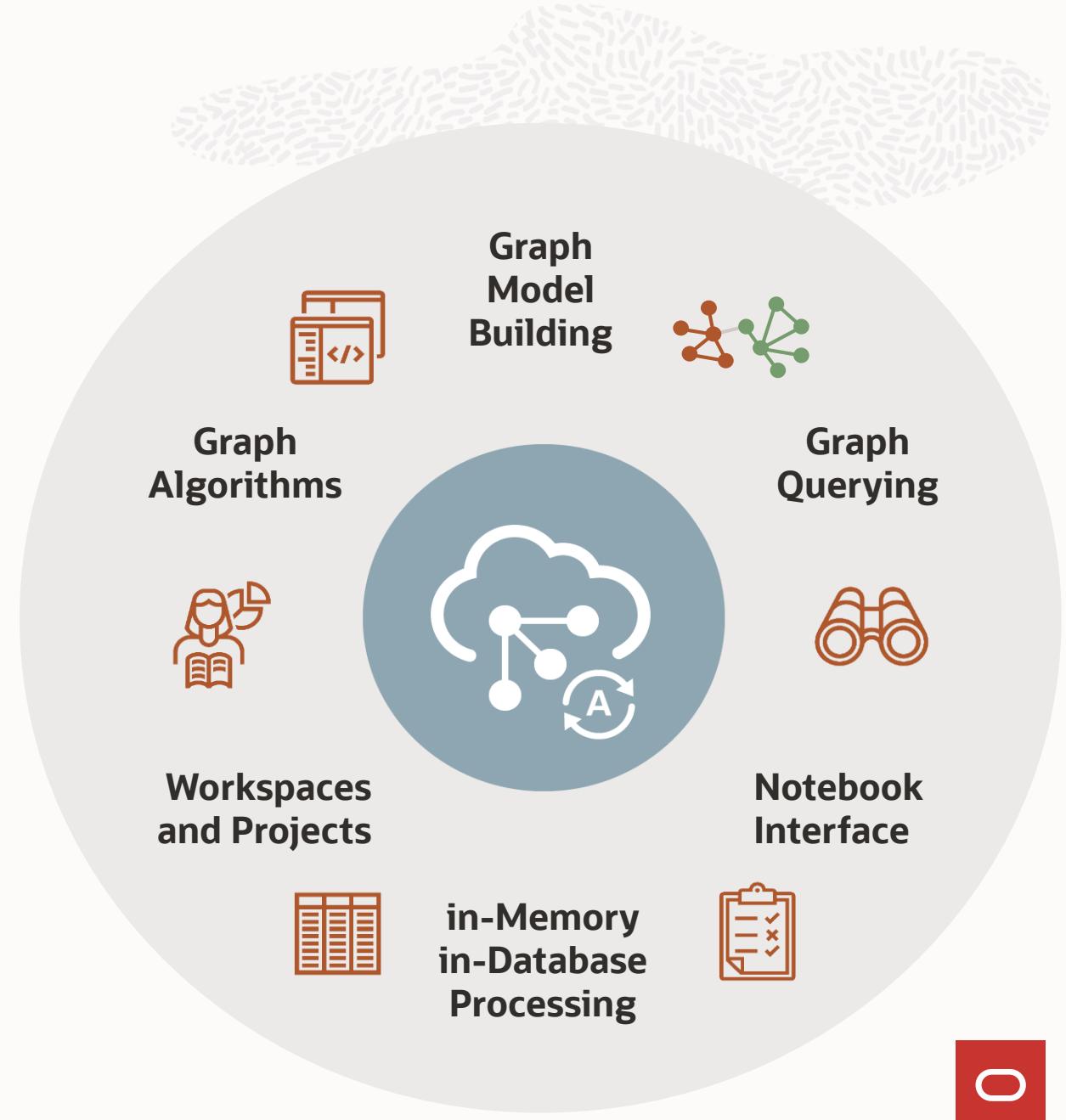
Model entities and relationships using a graph approach

Leverage comprehensive tooling and automation for graph modelling

Benefit from 50+ pre-built graph algorithms, graph querying language and visualization layer

Collaborate with teammates on shareable and reproducible graph assets

Run scalable and highly performant workloads on a managed environment



Oracle Machine Learning

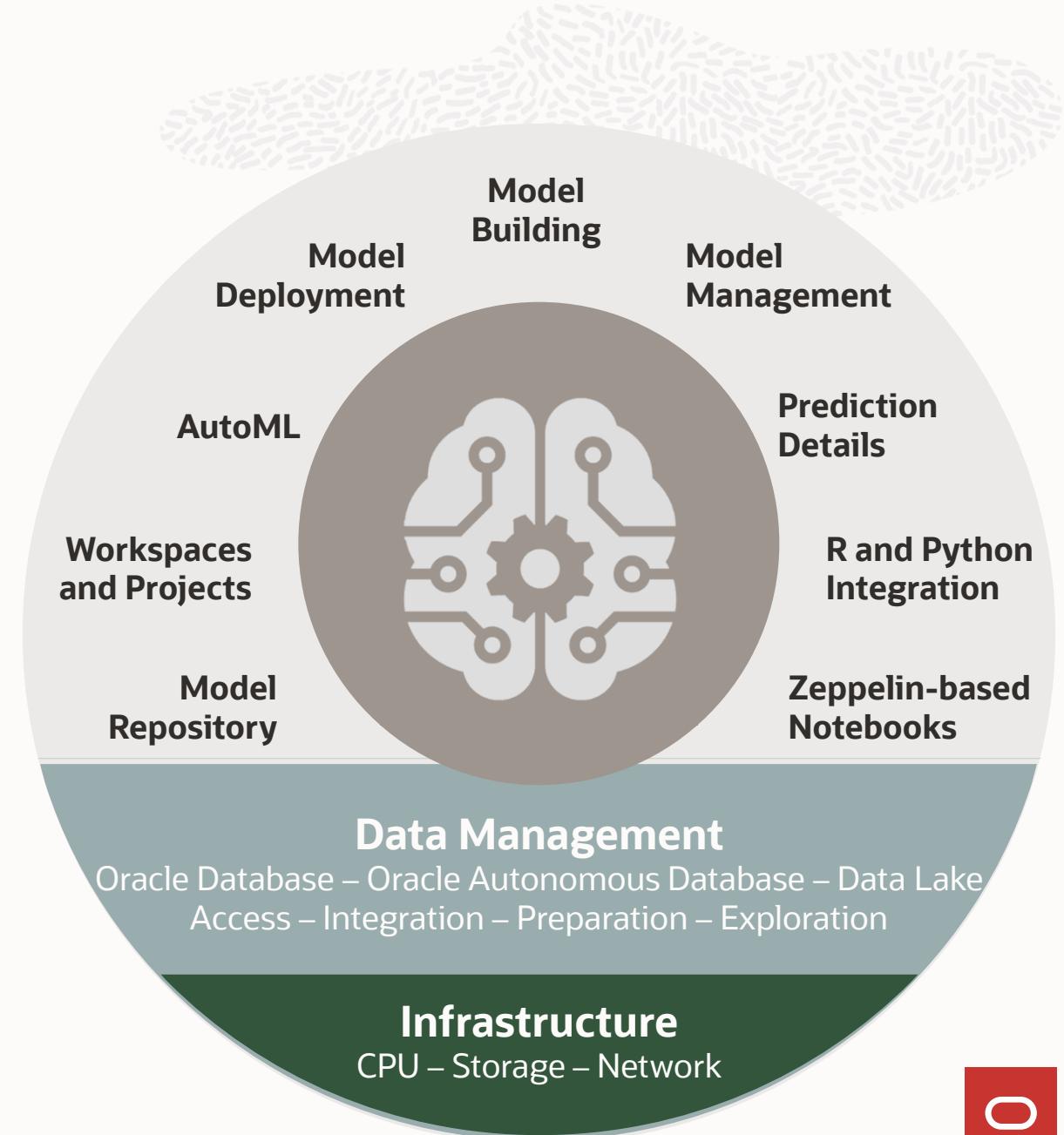
Support data scientist productivity and empower non-experts with AutoML

Gain algorithm-specific data preparation, integrated text mining, and partitioned models

Benefit from over 30+ high performance in-database machine learning algorithms

Deploy and update machine learning models in production via SQL and REST APIs

Deploy R and Python user-defined functions using managed processes



Data Science Service

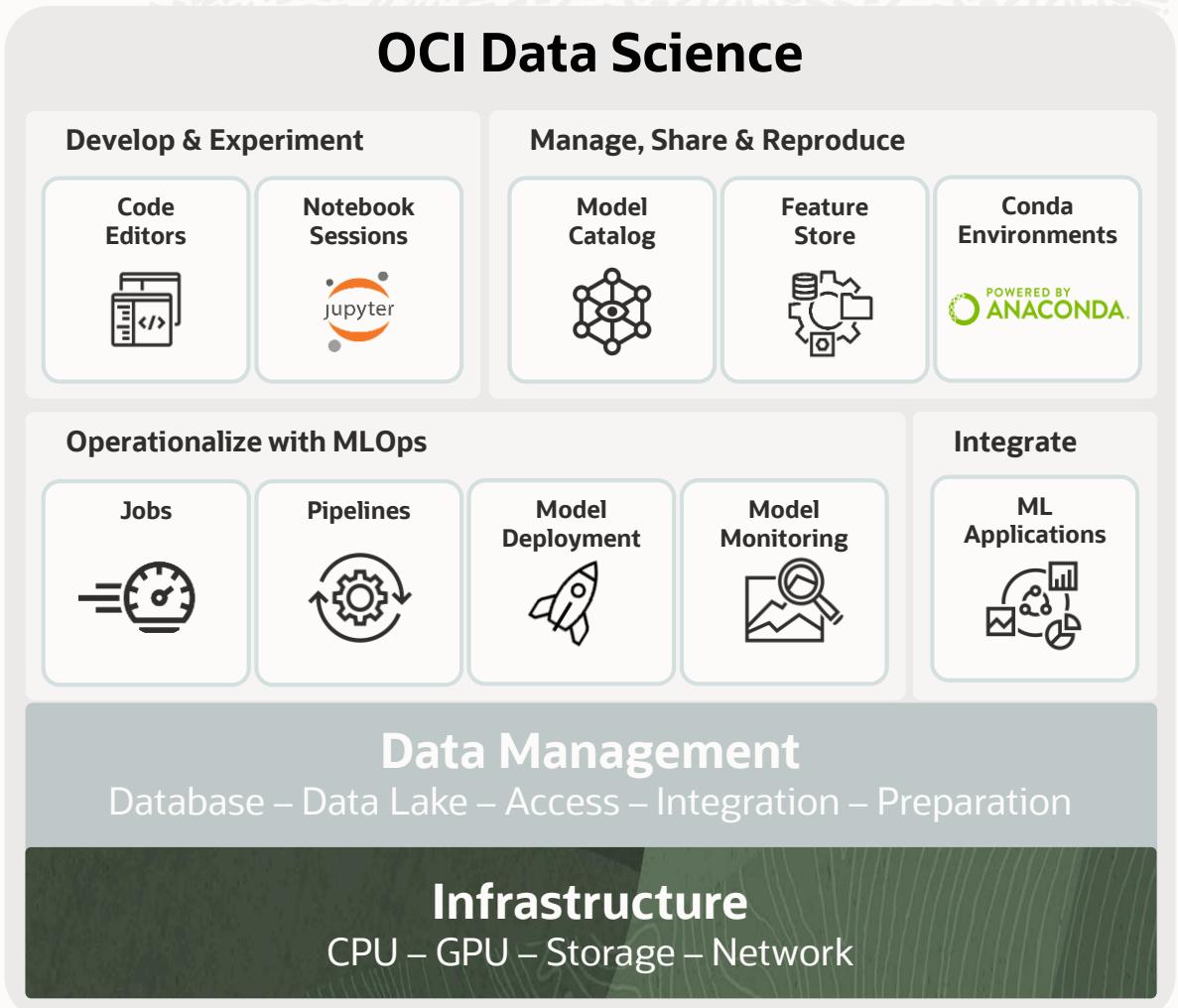
Accelerate and automate the entire end-to-end data science lifecycle

Use favorite open-source Python tools and frameworks

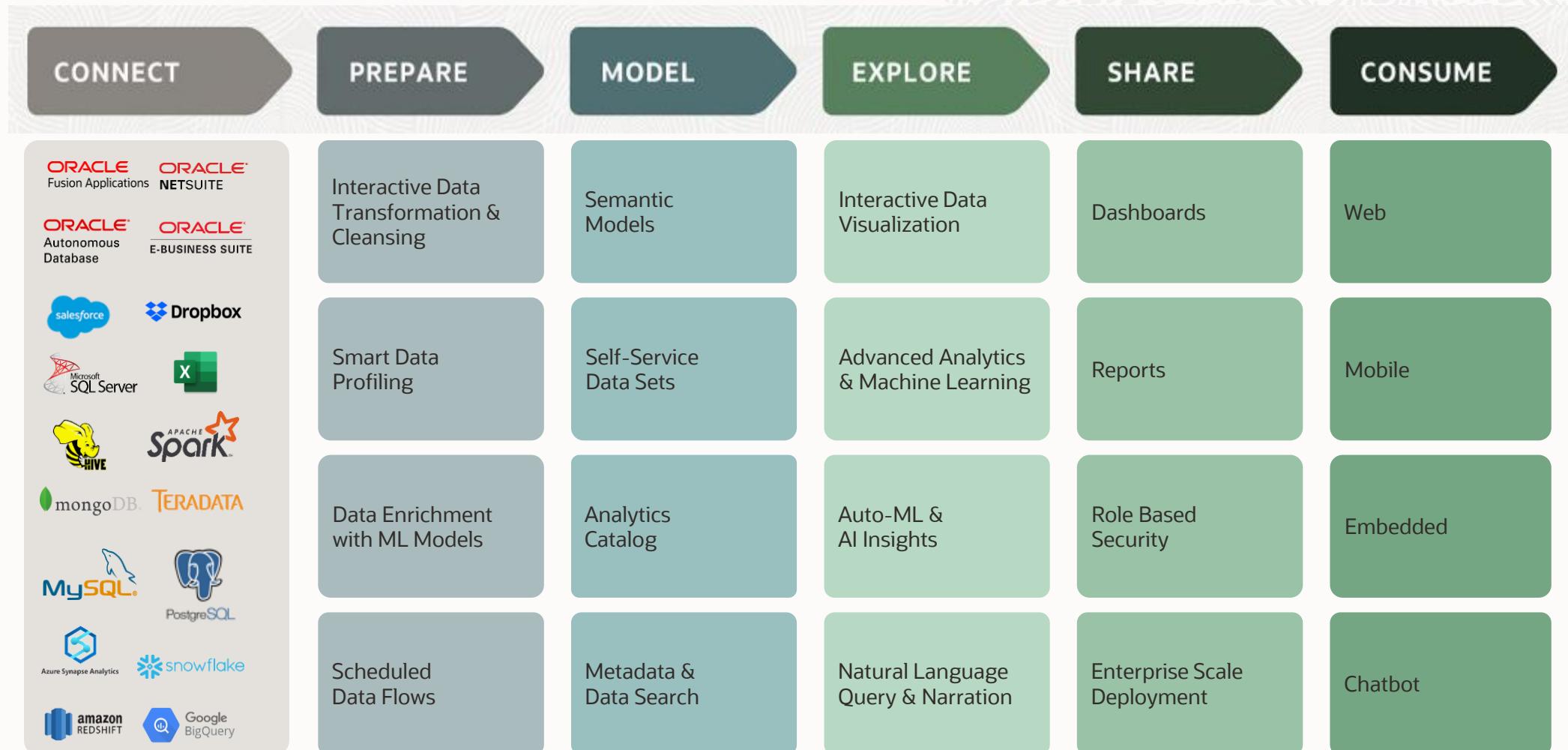
Gain enterprise-grade MLOps with flexible interfaces and unlimited scale

Collaborate with teammates on shareable and reproducible data science assets

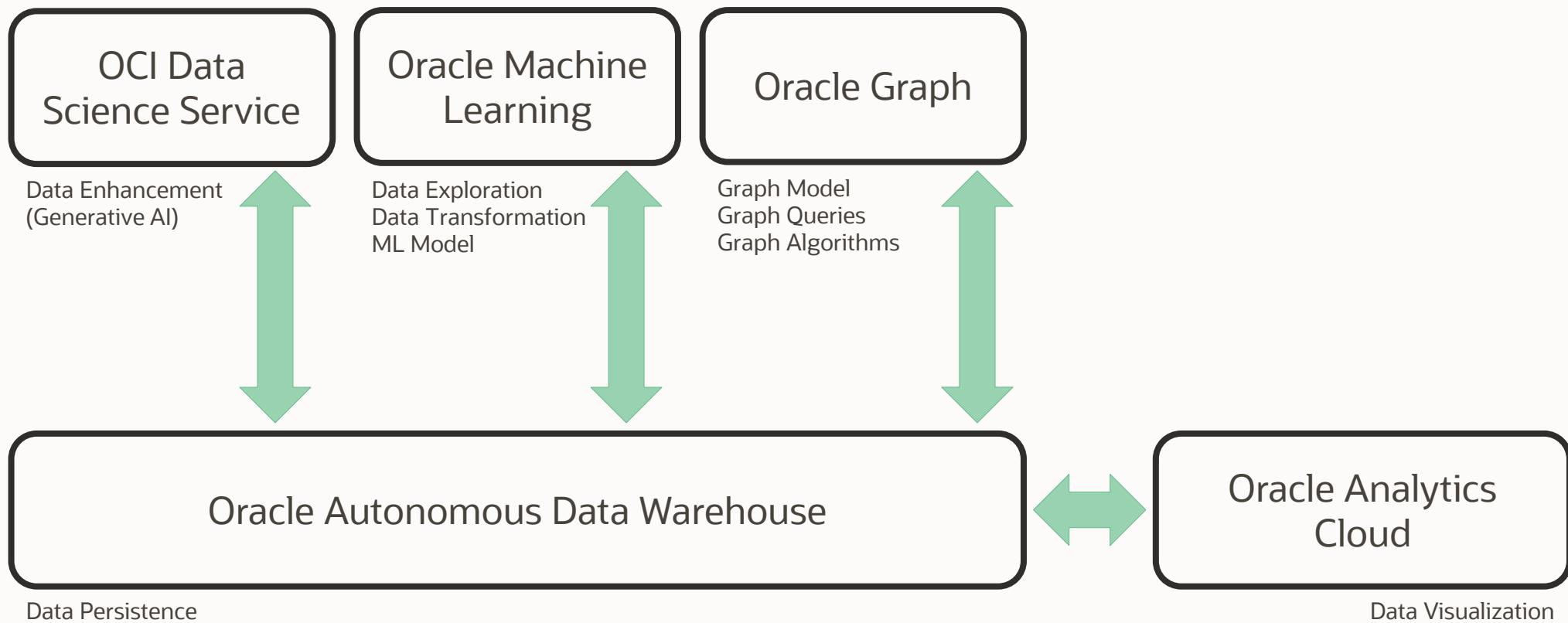
Run large-scale workloads with access to bare metal GPUs and distributed data processing and model training



Oracle Analytics Cloud

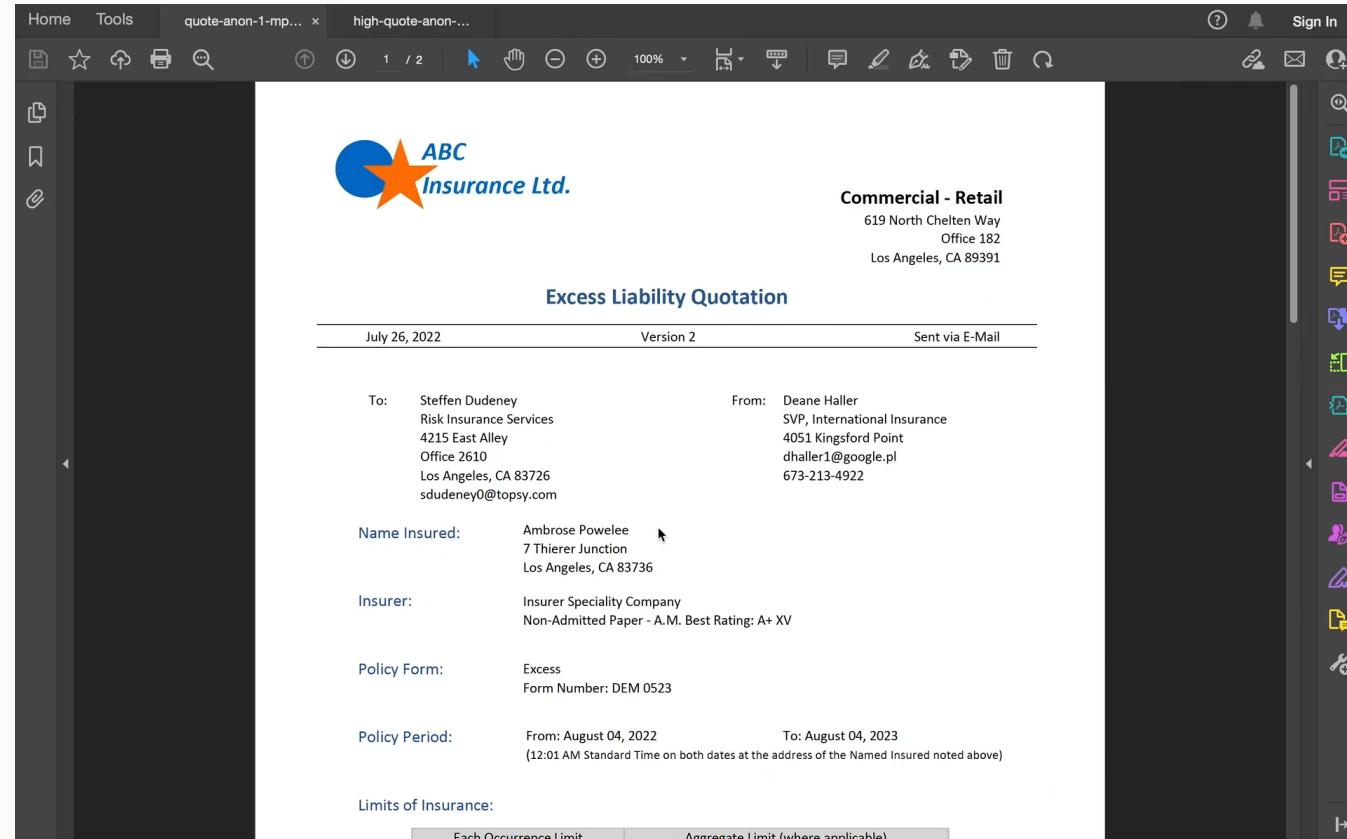


High Level Architecture



Other Potential Contributions

Discrepancies in VAT Forms using Document Understanding Service



- Compare the figures reported on VAT forms (e.g., VAT returns, invoices, sales records) with each other. Discrepancies, such as inconsistencies in the amounts reported, can be indicators of potential fraud.
- Investigate VAT refund claims that are significantly higher than usual.
- Find anomalies and irregularities in VAT forms
- ...

Social Fraud Detection with AI Language & AI Vision Service



An individual's Instagram profile is filled with photos of luxury vacations, high-end fashion, and expensive dining experiences. However, their VAT claim forms consistently report a very low income and minimal expenses

A business owner claims VAT refunds for international business travel expenses on their VAT return. However, their Instagram posts during the claimed travel period show them at a different location

A restaurant owner posts photos of a newly renovated, upscale restaurant on social media. However, their VAT returns consistently report low sales and expenses.

Thank you

EMEA Data Science Team

ORACLE

