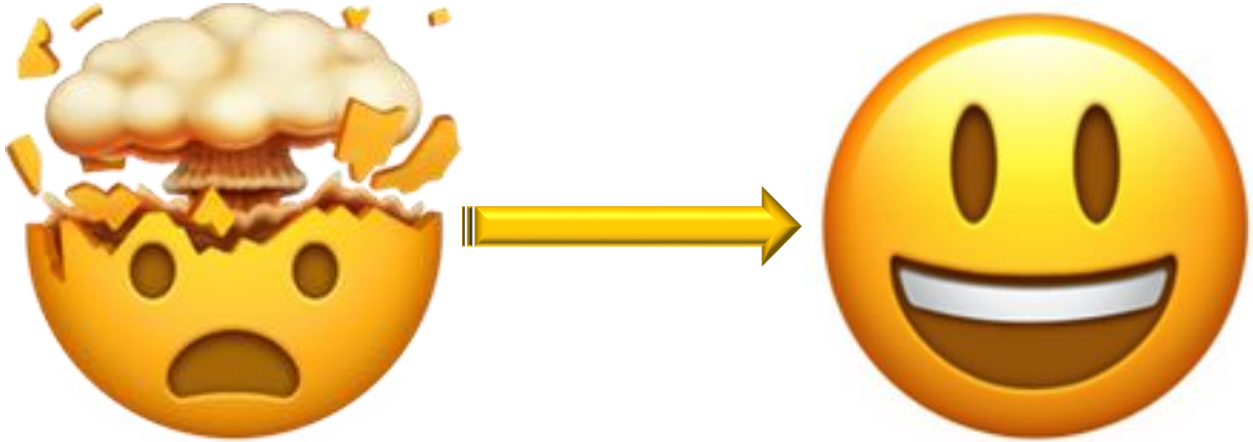
REST-Enabled Neural Networks in Oracle 18c



Brendan Tierney

ORALYTICS

Neil Chandler

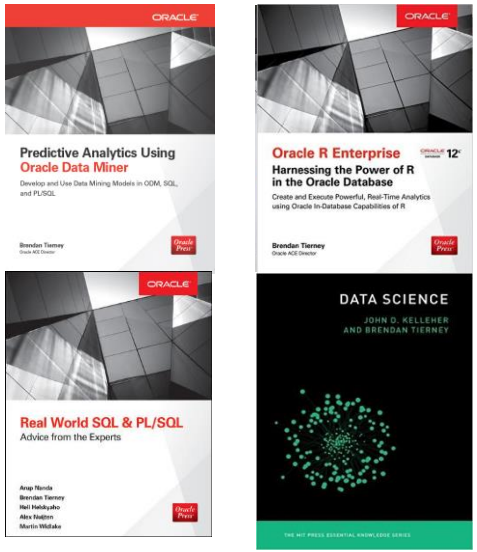
Brendan Tierney

Oracle Developer Champion

Oracle ACE Director

Oracle Groundbreaker Ambassador

- Data Warehousing since 1997
- Data Mining since 1998
- Analytics since 1993

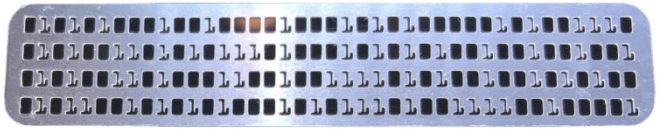


UK OUG
UK ORACLE USER GROUP

Neil Chandler

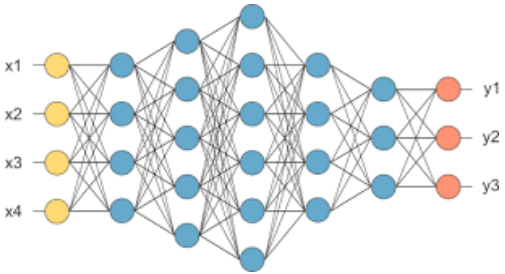
Oracle ACE Director

- IT since 1988
- Oracle since 1991 (v6)
- Developer and DBA



- @chandlerDBA
- http://chandlerDBA.com

WHAT



Portuguese Customer Marketing data set

Want to use machine learning to predict what customers are likely to open a new type of bank account.

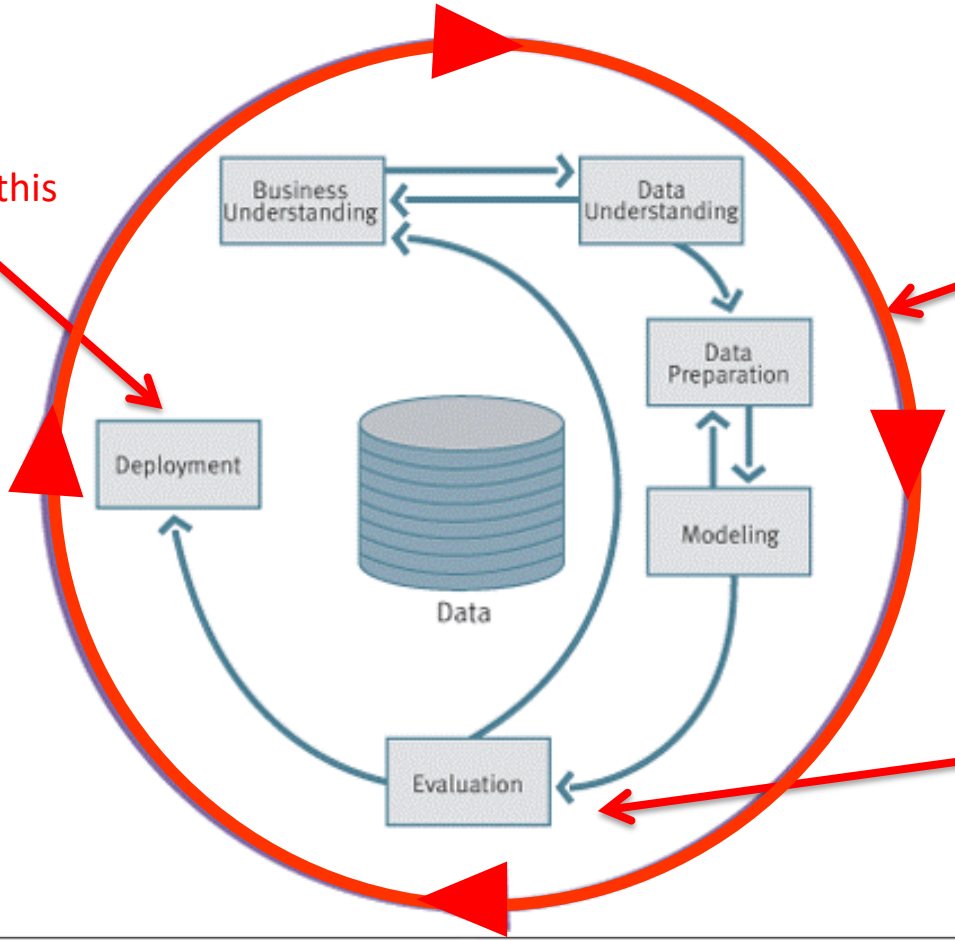
https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

WHY

MACHINE LEARNING

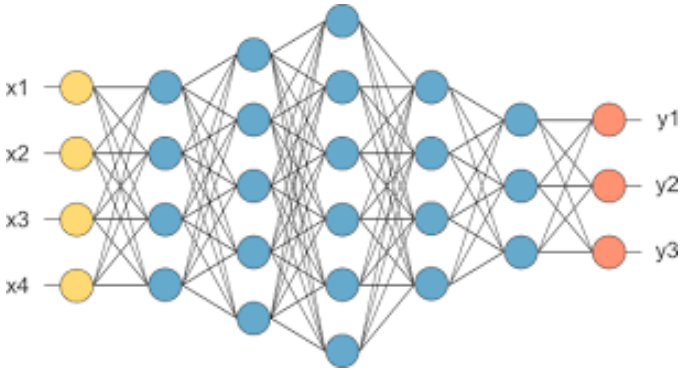
Nobody talks about this

The forgotten part / The part that is not talked about or discussed

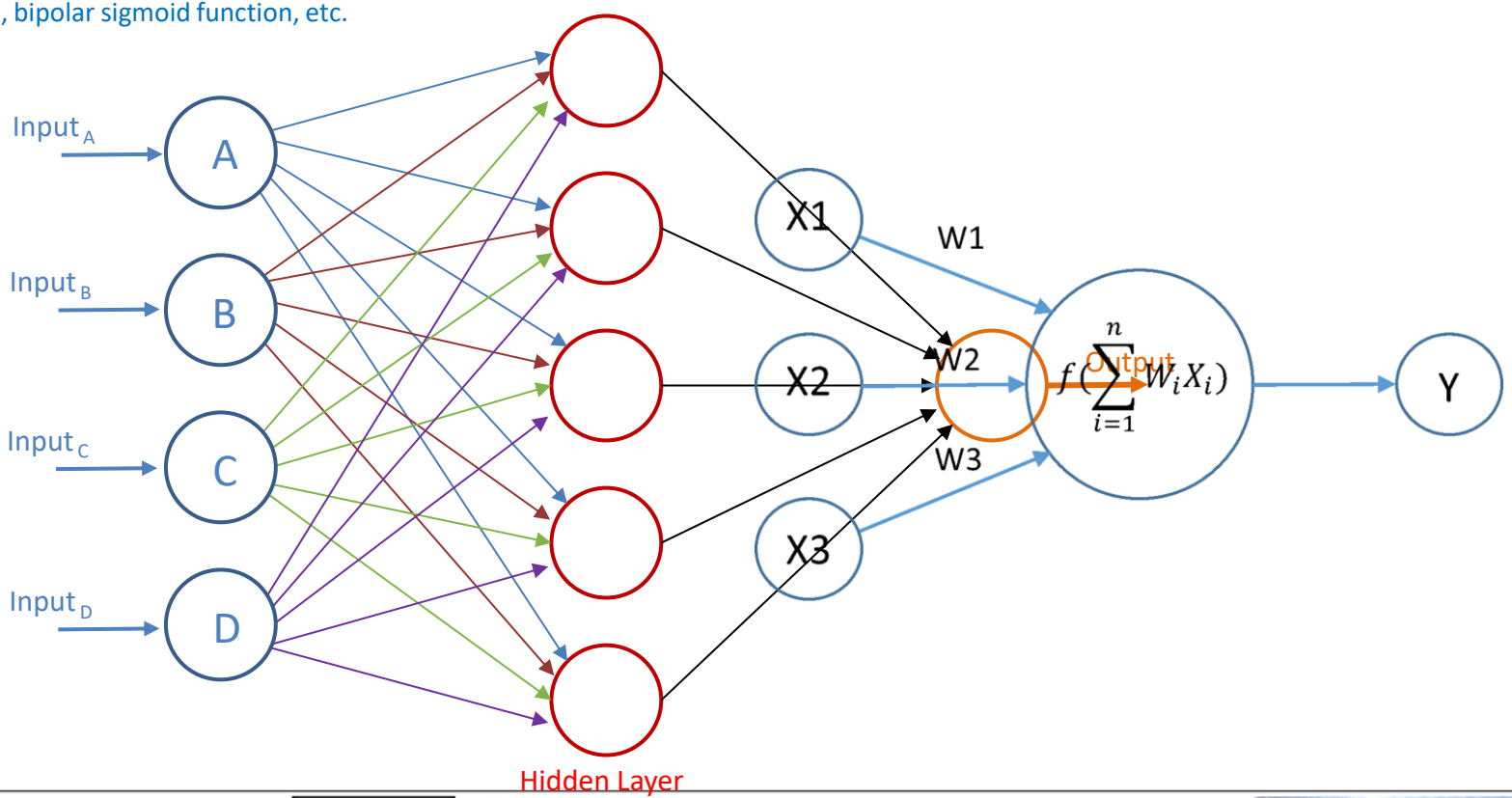


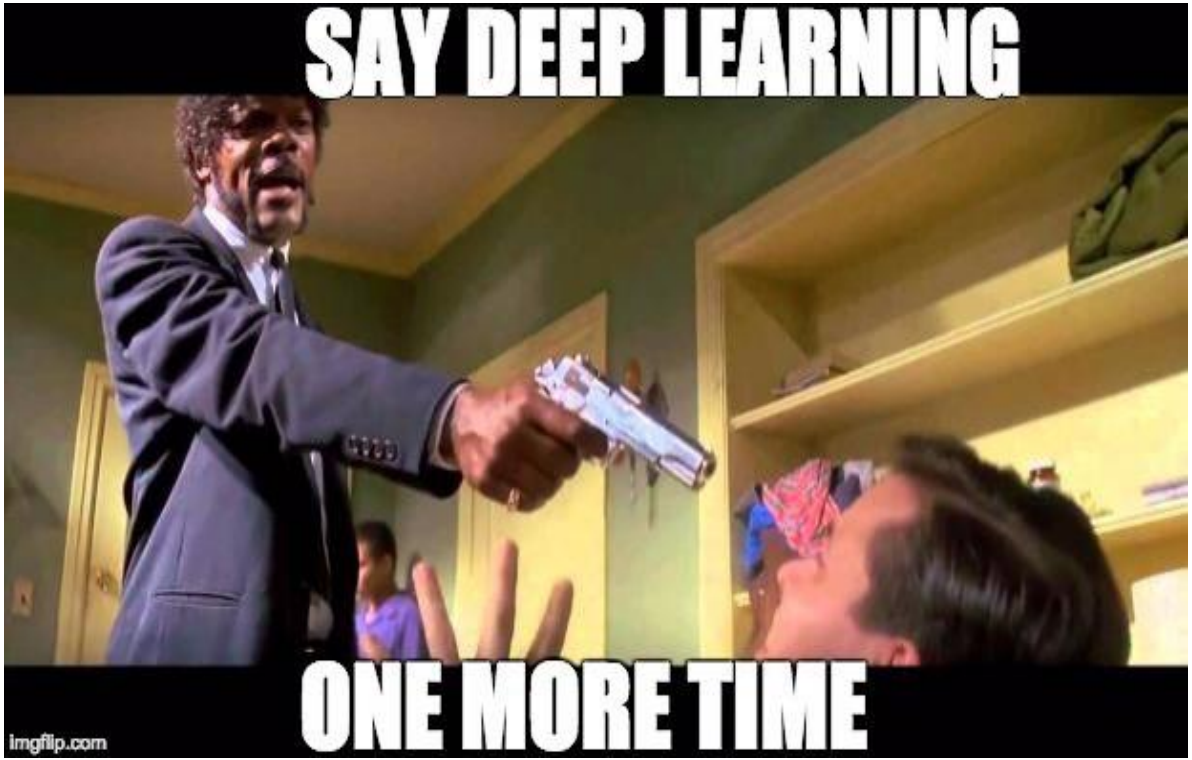
Everyone only talks up to this point

HOW


x1
x2
x3
x4
y1
y2
y3

- All the inputs are connected each neuron in the hidden layer (red circles).
- Weights are applied to each input to a neuron
- A neuron takes a set of numeric values as input and maps them to a single output value.
- A neuron is a simple multi-input linear regression function, where the output is passed through an activation function.
  - Two common activations functions are logistic and tanh functions. There are many others including logistic sigmoid function, arctan function, bipolar sigmoid function, etc.

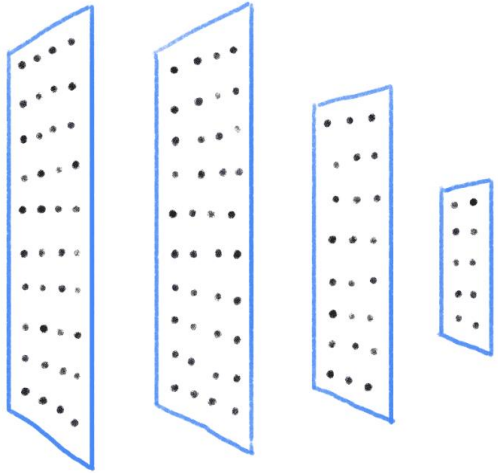



SAY DEEP LEARNING
ONE MORE TIME
imgflip.com

CAT

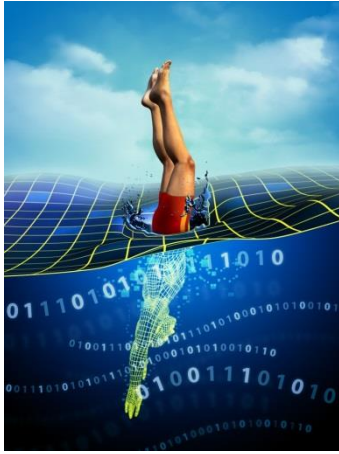
(LABELED PHOTOS)

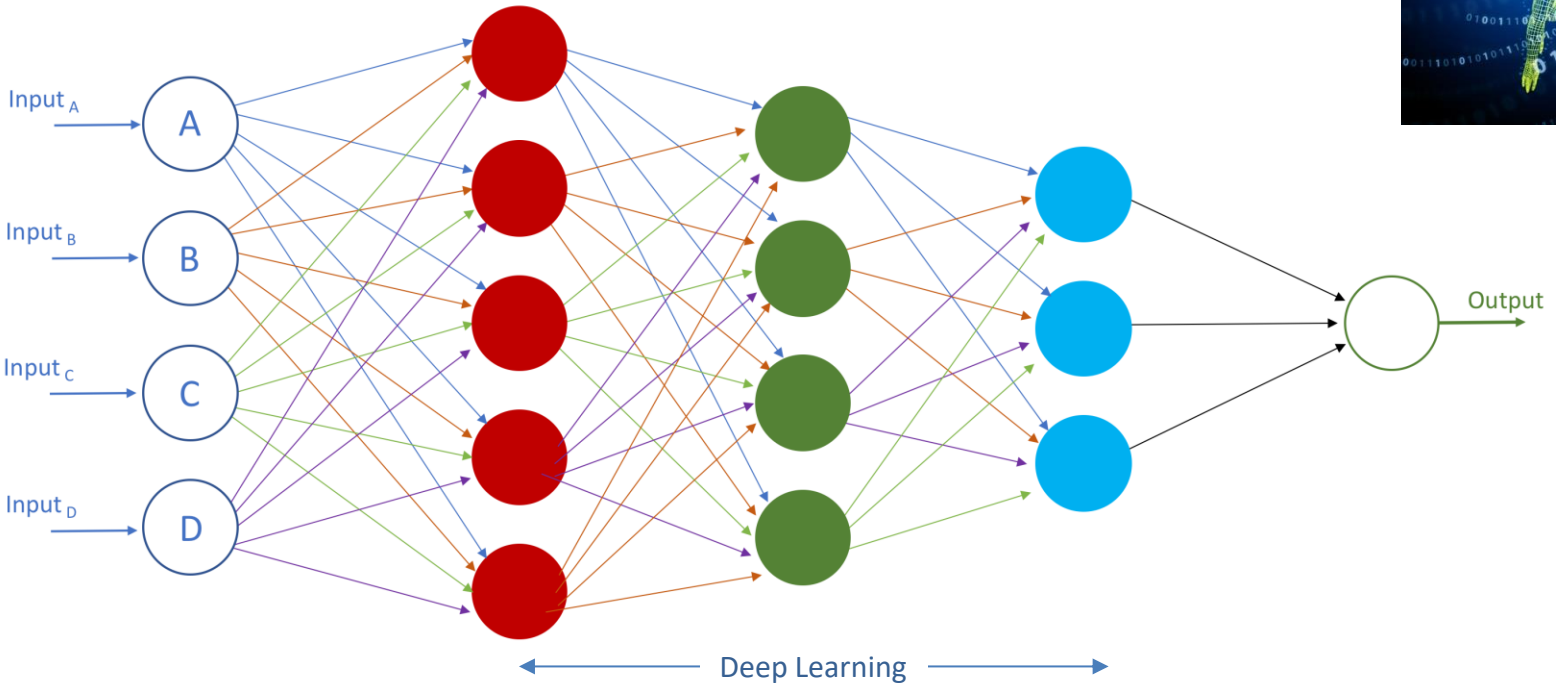
DOG



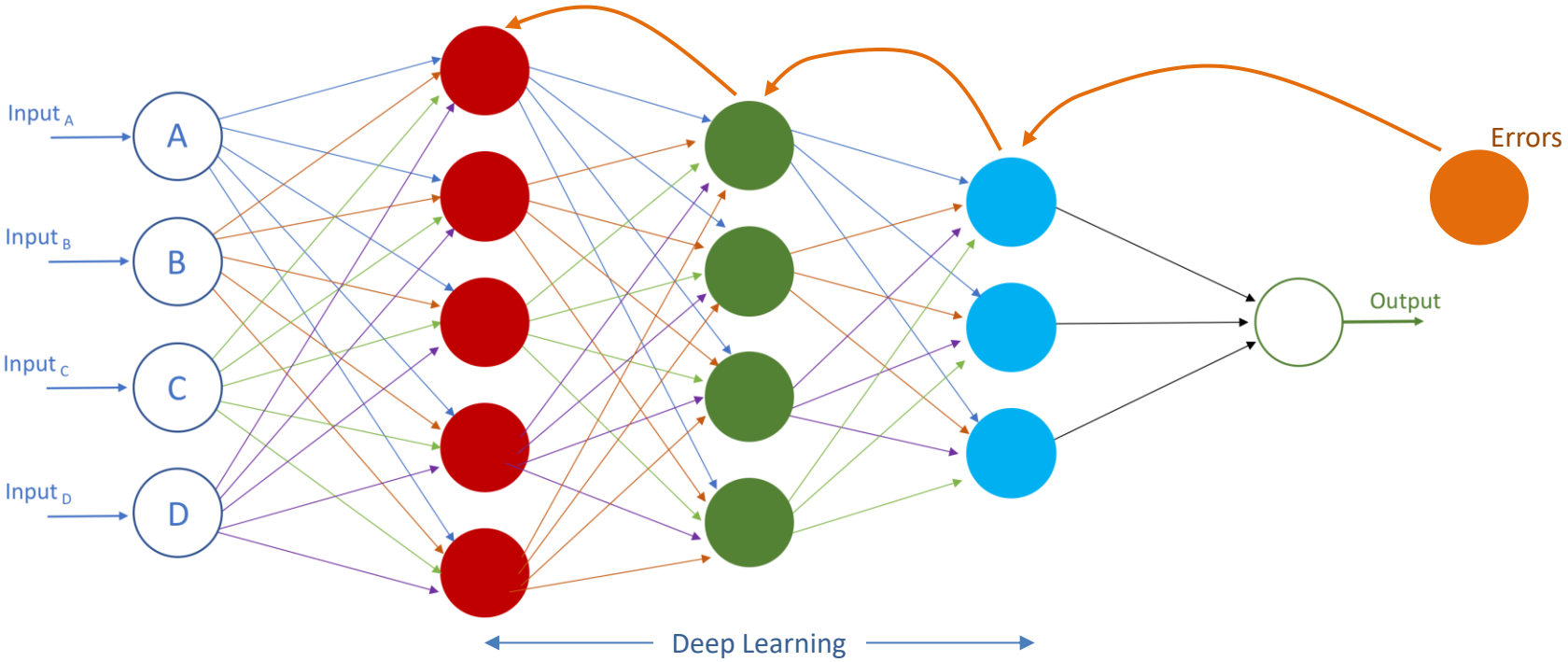
OUTPUT

- Additional hidden layers can be added
- The complexity of the computations involved can increases dramatically.
- But the addition of additional hidden layers gives the Neural Network to find additional and at time very complex patterns hidden in the data.
  - Deep Learning
- Each hidden layer finds deeper connections between the different input features
- More hidden layers => more complex processing and time





- Back Propagation Neural Networks
  - Training Data Set and Error rates are fed into the network (hidden layers)
  - The processing at each neuron is updated and the weights rules adjusted to reduce the error rate.
  - The Neural Network model starts by assuming random weights to each of the connections in the network. The algorithm then iteratively updates the weights in the network by showing training data to the network and updating the network weights until the network is optimized, in a generalized way to avoid over fitting.
    - Neural Networks learn and then relearn/update themselves



HOW

do we create a Neural Network Model?

Oracle Machine Learning Algorithms

Classification

- Naïve Bayes
- Logistic Regression (GLM)
- Decision Tree
- Random Forest
- Neural Network
- Support Vector Machine
- Explicit Semantic Analysis

Clustering

- Hierarchical K-Means
- Hierarchical O-Cluster
- Expectation Maximization (EM)

Anomaly Detection

- One-Class SVM

Time Series

- State of the art forecasting using Exponential Smoothing.
- Includes all popular models e.g. Holt-Winters with trends, seasons, irregularity, missing data

Regression

- Linear Model
- Generalized Linear Model
- Support Vector Machine (SVM)
- Stepwise Linear regression
- Neural Network
- LASSO

Attribute Importance

- Minimum Description Length
- Principal Comp Analysis (PCA)
- Unsupervised Pair-wise KL Div
- CUR decomposition for row & AI

Association Rules

- A priori/ market basket

Predictive Queries

- Predict, cluster, detect, features

SQL Analytics

- SQL Windows, SQL Patterns, SQL Aggregates

Feature Extraction

- Principal Comp Analysis (PCA)
- Non-negative Matrix Factorization
- Singular Value Decomposition (SVD)
- Explicit Semantic Analysis (ESA)

Text Mining Support

- Algorithms support text type
- Tokenization and theme extraction
- Explicit Semantic Analysis (ESA) for document similarity

Statistical Functions

- Basic statistics: min, max, median, stdev, t-test, F-test, Pearson's, Chi-Sq, ANOVA, etc.

R Packages

- CRAN R Algorithm Packages through Embedded R Execution
- Spark MLlib algorithm integration

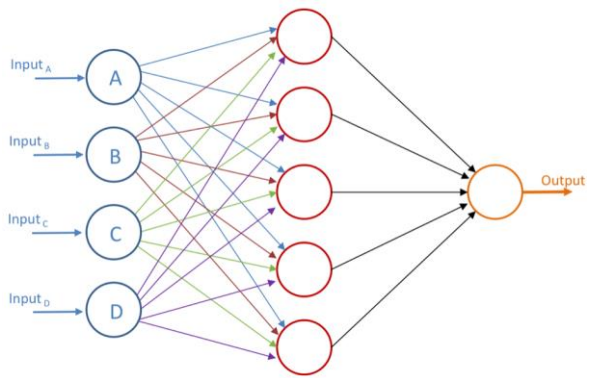
Exportable ML Models

- REST APIs for deployment

```
drop table demo_nnet_settings_default;
CREATE TABLE demo_nnet_settings_default
( setting_name  VARCHAR2(30),
  setting_value VARCHAR2(4000));

-- insert the settings records for a Neural Network
-- ADP is turned on. By default ADP is turned off.
BEGIN
  INSERT INTO demo_nnet_settings_default (setting_name, setting_value)
  values (dbms_data_mining.algo_name, dbms_data_mining.algo_neural_network);

  INSERT INTO demo_nnet_settings_default (setting_name, setting_value)
  VALUES (dbms_data_mining.prep_auto, dbms_data_mining.prep_auto_on);
END;
```

	SETTING_NAME	SETTING_VALUE	SETTING_TYPE
1	ALGO_NAME	ALGO_NEURAL_NETWORK	INPUT
2	NNET_HIDDEN_LAYERS	1	DEFAULT
3	NNET_ITERATIONS	200	DEFAULT
4	PREP_AUTO	ON	INPUT
5	NNET_ACTIVATIONS	'NNET_ACTIVATIONS_LOG_SIG'	DEFAULT
6	ODMS_RANDOM_SEED	0	DEFAULT
7	NNET_HELDASIDE_RATIO	.25	DEFAULT
8	NNET_HELDASIDE_MAX_FAIL	6	DEFAULT
9	LBFGS_HISTORY_DEPTH	20	DEFAULT
10	CLAS_WEIGHTS_BALANCED	OFF	DEFAULT
11	NNET_TOLERANCE	.000001	DEFAULT
12	ODMS_DETAILS	ODMS_ENABLE	DEFAULT
13	LBFGS_SCALE_HESSIAN	LBFGS_SCALE_HESSIAN_ENABLE	DEFAULT
14	LBFGS_GRADIENT_TOLERANCE	.000000001	DEFAULT
15	ODMS_MISSING_VALUE_TREATMENT	ODMS_MISSING_VALUE_AUTO	DEFAULT
16	ODMS_SAMPLING	ODMS_SAMPLING_DISABLE	DEFAULT

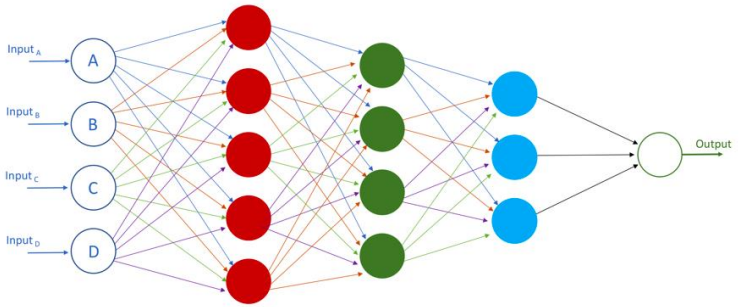
```
BEGIN
   DBMS_DATA_MINING.CREATE_MODEL(
      model_name          => 'DEMO_NNET_MODEL_1',
      mining_function     => dbms_data_mining.classification,
      data_table_name     => 'banking_marketing_Training_v',
      case_id_column_name => 'ID',
      target_column_name  => 'TARGET',
      settings_table_name => 'demo_nnet_settings_default');
END;
```



```
drop table demo_nnet_settings_default_2;
CREATE TABLE demo_nnet_settings_default_2
( setting_name  VARCHAR2(30),
  setting_value VARCHAR2(4000));

BEGIN
  INSERT INTO demo_nnet_settings_default_2 (setting_name, set
  values (dbms_data_mining.algo_name, dbms_data_mining.algo_n

  INSERT INTO demo_nnet_settings_default_2 (setting_name, set
  VALUES (dbms_data_mining.prep_auto, dbms_data_mining.prep_a

  INSERT INTO demo_nnet_settings_default_2 (setting_name, set
  VALUES (dbms_data_mining.nnet_nodes_per_layer, '20,10,6');
END;
```

	SETTING_NAME	SETTING_VALUE	SETTING_TYPE
1	ALGO_NAME	ALGO_NEURAL_NETWORK	INPUT
2	NNET_HIDDEN_LAYERS	3	DEFAULT
3	NNET_ITERATIONS	200	DEFAULT
4	PREP_AUTO	ON	INPUT
5	NNET_ACTIVATIONS	'NNET_ACTIVATIONS_LOG_SIG','NNET_ACTIVATIONS_LOG_SIG','NNET_ACTIVATIONS_LOG_SIG'	DEFAULT
6	ODMS_RANDOM_SEED	0	DEFAULT
7	NNET_HELDASIDE_RATIO	.25	DEFAULT
8	NNET_HELDASIDE_MAX_FAIL	6	DEFAULT
9	LBFGS_HISTORY_DEPTH	20	DEFAULT
10	CLAS_WEIGHTS_BALANCED	OFF	DEFAULT
11	NNET_TOLERANCE	.000001	DEFAULT
12	NNET_NODES_PER_LAYER	20,10,6	INPUT
13	ODMS_DETAILS	ODMS_ENABLE	DEFAULT
14	LBFGS_SCALE_HESSIAN	LBFGS_SCALE_HESSIAN_ENABLE	DEFAULT
15	LBFGS_GRADIENT_TOLERANCE	.000000001	DEFAULT
16	ODMS_MISSING_VALUE_TREATMENT	ODMS_MISSING_VALUE_AUTO	DEFAULT
17	ODMS_SAMPLING	ODMS_SAMPLING_DISABLE	DEFAULT

```
BEGIN
   DBMS_DATA_MINING.CREATE_MODEL(
      model_name          => 'DEMO_NNET_MODEL_3',
      mining_function     => dbms_data_mining.classification,
      data_table_name     => 'banking_marketing_Training_v',
      case_id_column_name => 'ID',
      target_column_name  => 'TARGET',
      settings_table_name => 'demo_nnet_settings_default');
END;
```



HOW

good is it?

Testing & Evaluating

- Other Models
  - Decision Trees, Naïve Bayes, GLM, SVM

Algorithm	Accuracy	Precision	Recall
GLM	83%	86%	41%
Decision Tree	82%	88%	40%
Naïve Bayes	80%	64%	34%
SVM	82%	88%	39%

	0 (condition negative)	1 (condition positive)	
0 (test outcome negative)	True Negative	False Negative (Type II Errors)	**Negative Prediction Rate** = ∑True Negative / ∑ Total Negative
1 (test outcome positive)	False Positive (Type I Errors)	True Positive	**Precision** = Positive Prediction Rate = ∑True Positive / ∑Total Positive

Negative Rate = {∑False Negative + ∑False Positive} / ∑Total Population			**Accuracy** = {∑True Negative + ∑True Positive} / ∑Total Population
	True Negative Rate = **Specificity** = ∑True Negative / ∑All Negative	True Positive Rate = **Sensitivity = Recall** = ∑True Positive / ∑All Positive	

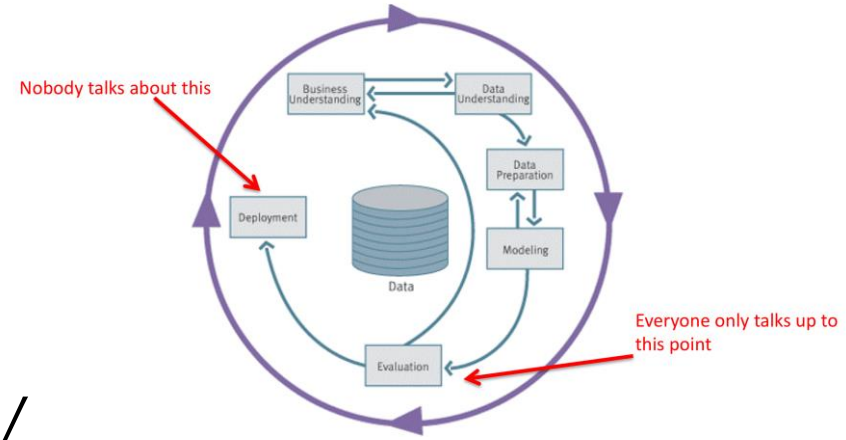
Testing & Evaluating

- Other Models
  - Decision Trees, Naïve Bayes, GLM, SVM

Algorithm	Accuracy	Precision	Recall
GLM	83%	86%	41%
Decision Tree	82%	88%	40%
Naïve Bayes	80%	64%	34%
SVM	82%	88%	39%
Neural Network 1	90%	54%	58%
Neural Network 2	92%	96%	89%
Neural Network 3	93%	99%	88%

	0 (condition negative)	1 (condition positive)	
0 (test outcome negative)	True Negative	False Negative (Type II Errors)	**Negative Prediction Rate** = ∑True Negative / ∑ Total Negative
1 (test outcome positive)	False Positive (Type I Errors)	True Positive	**Precision** = Positive Prediction Rate = ∑True Positive / ∑Total Positive
Negative Rate = {∑False Negative + ∑False Positive} / ∑Total Population			**Accuracy** = {∑True Negative + ∑True Positive} / ∑Total Population
	True Negative Rate = **Specificity** = ∑True Negative / ∑All Negative	True Positive Rate = **Sensitivity = Recall** = ∑True Positive / ∑All Positive	

20,10,6 20,8,4



HOW

do we use this Neural Network Model?

Making Predictions

```
-- Using the Neural Network to label data (make a prediction)

        PREDICTION
        PREDICTION_PROBABILITY
```

Making Predictions

```
-- Using the Neural Network to label data (make a prediction)

        PREDICTION(DEMO_NNET_MODEL_3 USING *)
        PREDICTION_PROBABILITY(DEMO_NNET_MODEL_3 USING *)
```

Making Predictions

```
-- Using the Neural Network to label data (make a prediction)

       PREDICTION(DEMO_NNET_MODEL_3 USING *)
       PREDICTION_PROBABILITY(DEMO_NNET_MODEL_3 USING *)
FROM banking_marketing_data
```

Making Predictions

```
-- Using the Neural Network to label data (make a prediction)
SELECT id, age, job, marital, target,
       PREDICTION(DEMO_NNET_MODEL_3 USING *) Predicted_Value,
       PREDICTION_PROBABILITY(DEMO_NNET_MODEL_3 USING *) Prob
FROM banking_marketing_data
WHERE  rownum <= 8;
```

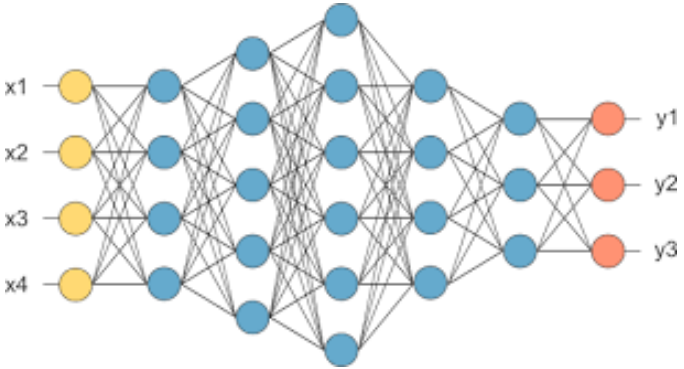
	ID	AGE	JOB	MARITAL	TARGET	PREDICTED_VALUE	PROB
1	AAATBVAAMAAAAEzAAA	30	technician	married	no	no	0.9998720931645994
2	AAATBVAAMAAAAEzAAB	33	services	divorced	no	no	0.9998586131862865
3	AAATBVAAMAAAAEzAAC	32	technician	single	no	no	0.9998648012493204
4	AAATBVAAMAAAAEzAAD	36	technician	single	no	no	0.9998630908847513
5	AAATBVAAMAAAAEzAAE	36	blue-collar	single	no	no	0.999690048182365
6	AAATBVAAMAAAAEzAAF	33	services	single	no	no	0.5875531335343148
7	AAATBVAAMAAAAEzAAG	40	technician	divorced	no	no	0.999842670818056
8	AAATBVAAMAAAAEzAAH	41	blue-collar	married	no	no	0.9998319477050562

What-If Analysis

```
SELECT prediction(DEMO_NNET_MODEL_3
            USING 30 AS age,
             'single' AS marital,
        'technician' AS job,
             'yes'   AS housing,
 'university.degree' AS education)  Predicted_Value,

       prediction_probability(DEMO_NNET_MODEL_3, 'no'
           USING 30 AS age,
            'single' AS marital,
        'technician' AS job,
             'yes'   AS housing,
 'university.degree' AS education) Predicted_Prob_NO,

       prediction_probability(DEMO_NNET_MODEL_3, 'yes'
           USING 30 AS age,
            'single' AS marital,
        'technician' AS job,
             'yes'   AS housing,
 'university.degree' AS education) Predicted_Prob_YES
FROM dual;
```

Add more attributes.

Change the values
=> see what happens to predicted outcome


x1
x2
x3
x4
y1
y2
y3

HOW

How best to expose this analysis?

How best to expose this analysis?

What is REST?

REpresentational **S**tate **T**ransfer

A URI request will elicit a response with a payload

WHY REST?

It is **easy**!

4 verbs…

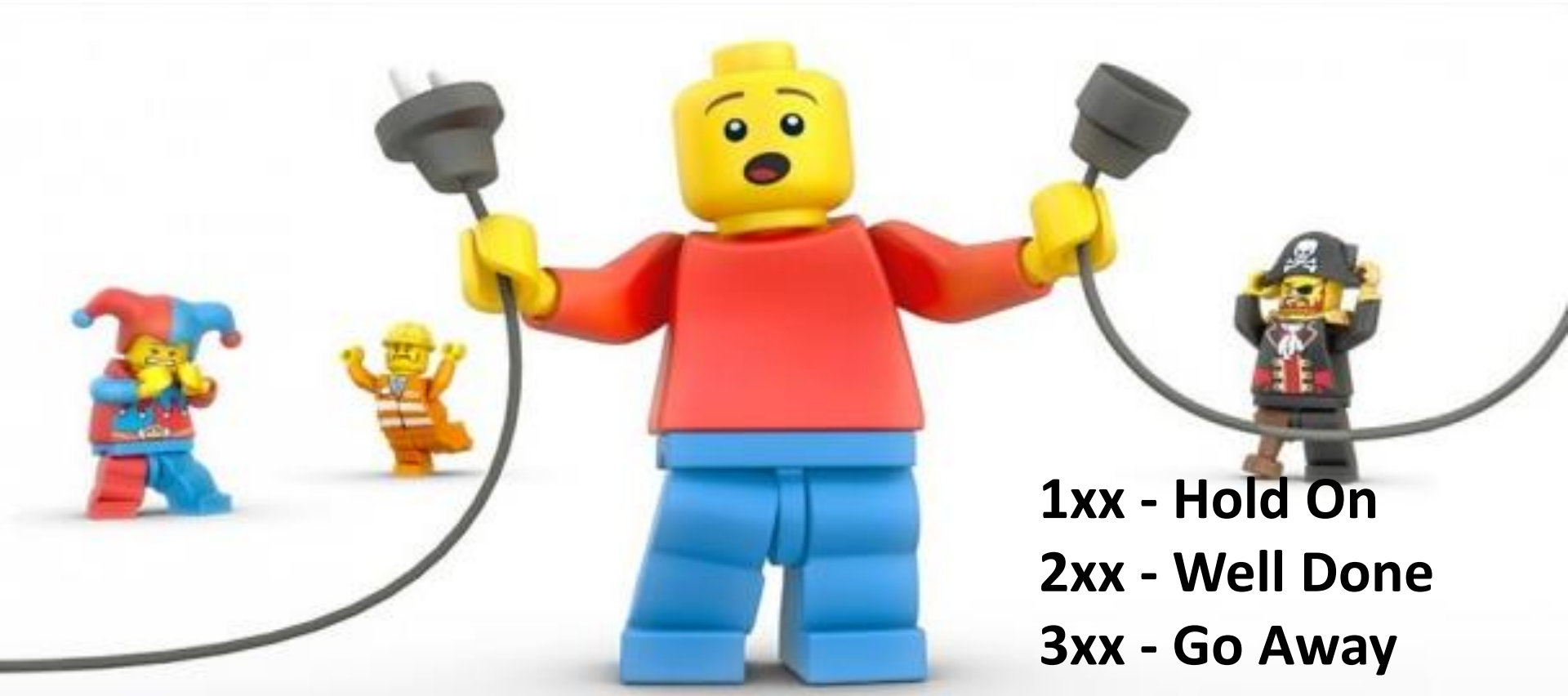
- GET (select)
- POST (update)
- PUT (insert)
- DELETE (delete)

REST Status codes

It is easy!

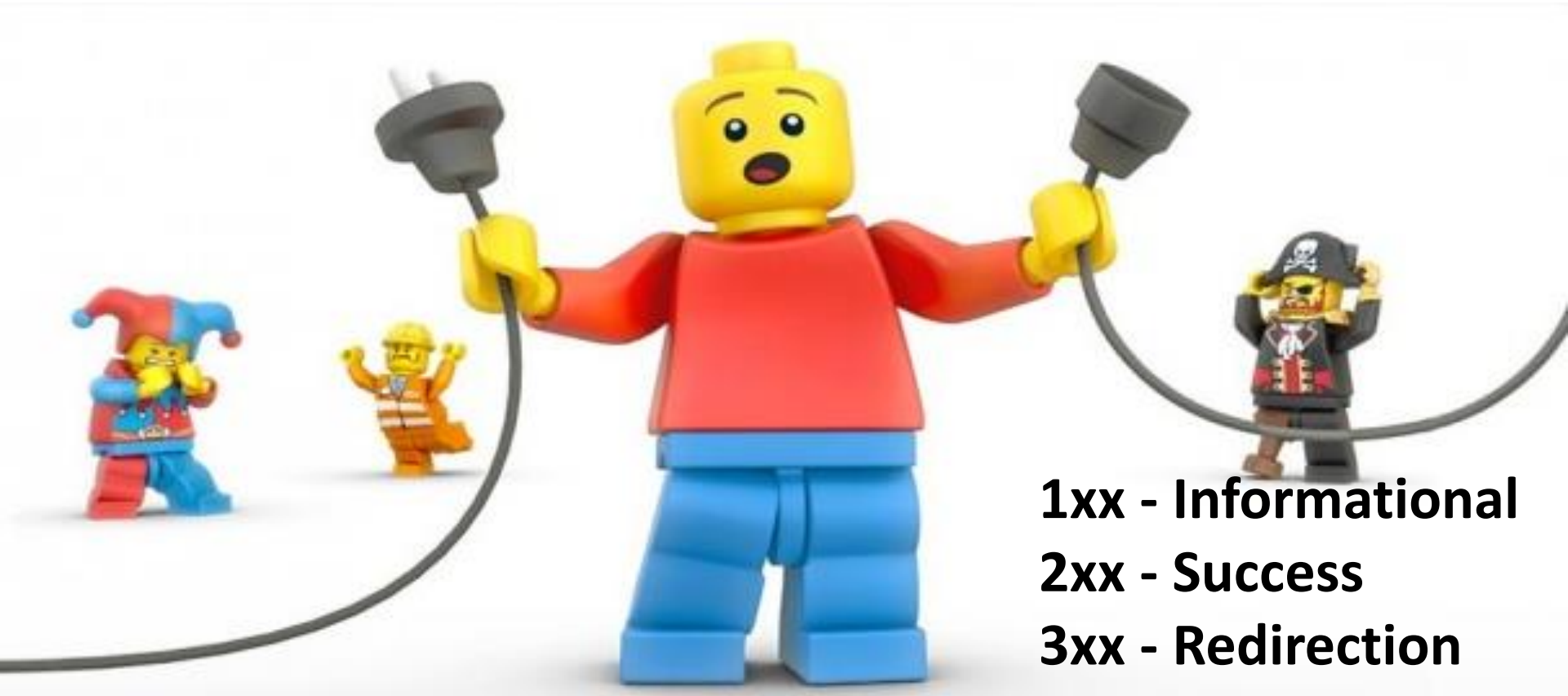
HTTP: 404

Not Found



REST Status Codes

1xx - Hold On
2xx - Well Done
3xx - Go Away
4xx - you messed up
5xx - I messed up



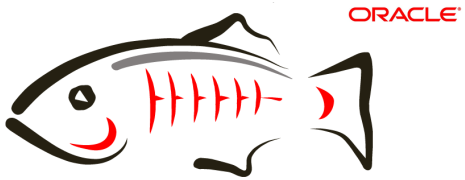
REST Status Codes

1xx - Informational
2xx - Success
3xx - Redirection
4xx - Client Error
5xx - Server Error

How do we do this with an Oracle database?

Oracle Rest Data Services - ORDS

Glassfish



Standalone/Jetty

Apache Tomcat

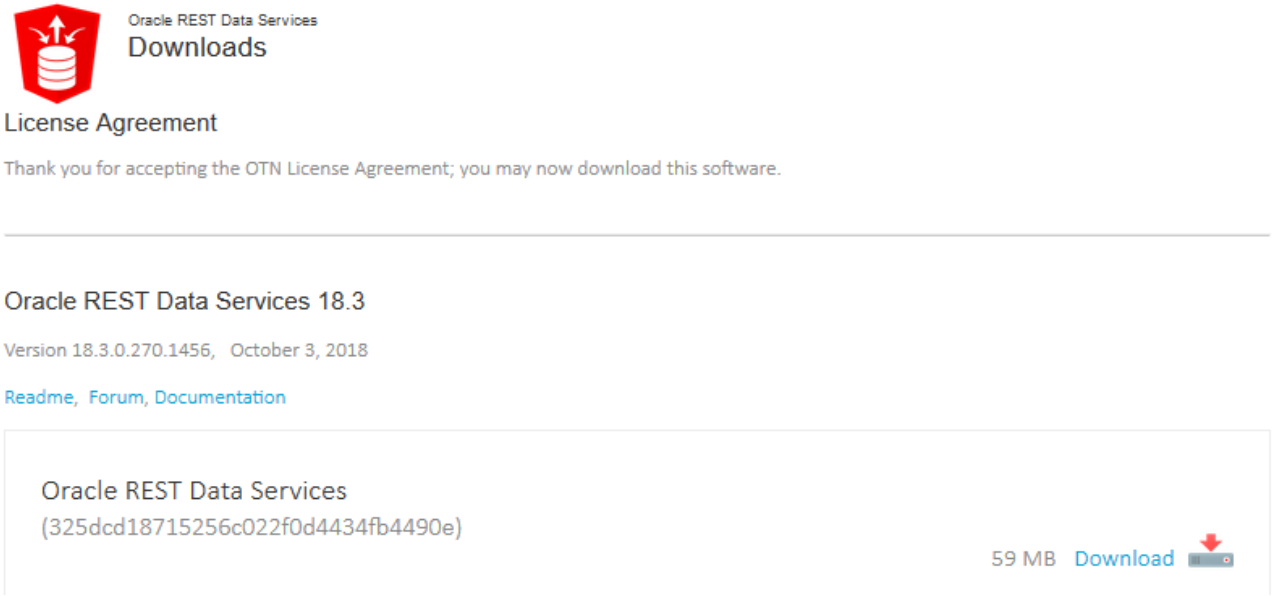
Weblogic

How do we do this with an Oracle database?

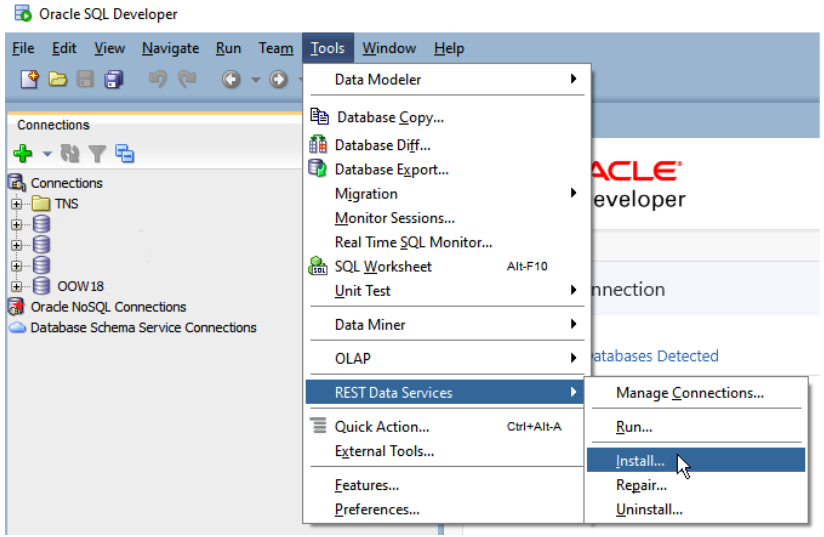
Oracle Rest Data Services - ORDS

Install

Download the latest version here:
https://www.oracle.com/technetwork/developer-tools/rest-data-services/downloads/index.html



Install ORDS



unzip the file you downloaded

```
$ java -version
java version "1.8.0_171"
```

```
$ cd /u01/app/oracle/product/ords
$ java -jar ords.war install
```

prompted install
(or you can pre-fill a response file)

You don't *need* APEX installed in the DB, but it's a good idea

Install ORDS

```
This Oracle REST Data Services instance has not yet been configured.
Please complete the following prompts

Enter the location to store configuration data:/u01/app/oracle/product/ords
Enter the name of the database server [localhost]: EE18AHP
Enter the database listen port [1521]: 1521
Enter the database service name: pdb1
Enter the database password for ORDS_PUBLIC_USER: passw0rd
Requires SYS AS SYSDBA to verify Oracle REST Data Services schema.
Enter the database password for SYS AS SYSDBA: change_on_install

Enter 1 if you want to use PL/SQL Gateway or 2 to skip this step.
If using Oracle Application Express or migrating from mod_plsql then you must enter 1 [1]:
Enter the database password for APEX_PUBLIC_USER: passw0rd
Enter the database password for APEX_LISTENER: passw0rd
Enter the database password for APEX_REST_PUBLIC_USER: passw0rd
.
Completed installation

And VALIDATE the install
$ java -jar ords.war validate
```

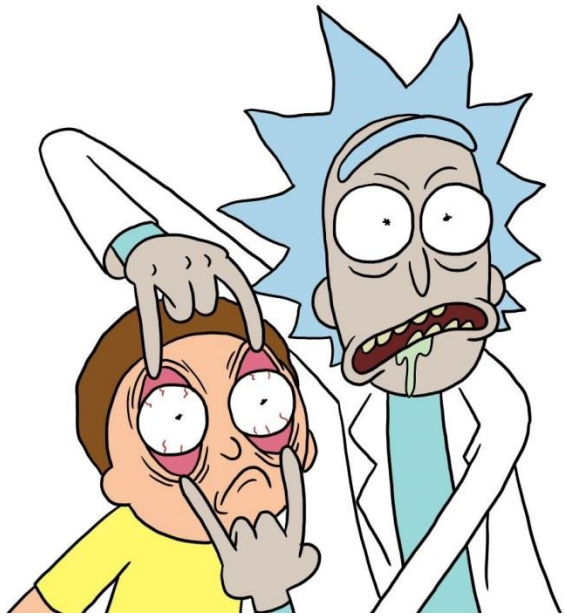

Start ORDS

```
JAVA_OPTIONS=
 "-Dorg.eclipse.jetty.server.Request.maxFormContentSize=2000000"

cd /u01/ords

$JAVA_HOME/bin/java ${JAVA_OPTIONS} \
                -jar ords.war standalone > ords.log 2>&1
```

URI: http://144.21.70.32:8080/**ords**

Simple Data Example

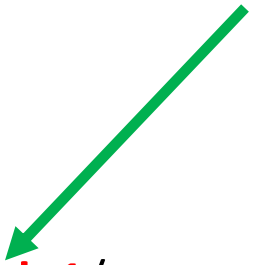
```
SELECT * FROM
 oow18.countries
 ORDER BY country_iso_code;
```



COUNTRY_ID	COUNTRY_ISO_CODE	COUNTRY_NAME
52773	AR	Argentina
52774	AU	Australia
52775	BR	Brazil
52772	CA	Canada
52771	CN	China
52776	DE	Germany
52777	DK	Denmark
52778	ES	Spain
52779	FR	France
52789	GB	United Kingdom
52780	IE	Ireland
52781	IN	India
52770	IT	Italy
52782	JP	Japan
52783	MY	Malaysia
52784	NL	The Netherlands
52785	NZ	New Zealand
52786	PL	Poland
52787	SA	Saudi Arabia
52769	SG	Singapore
52788	TR	Turkey
52790	US	United States of America
52791	ZA	South Africa

Enable REST for a schema

```
BEGIN
    ORDS.ENABLE_SCHEMA(
        p_enabled             => TRUE,
        p_schema              => 'OOW18',
        p_url_mapping_type    => 'BASE_PATH',
        p_auto_rest_auth      => FALSE,
        p_url_mapping_pattern => 'code1');
    commit;
END;
```



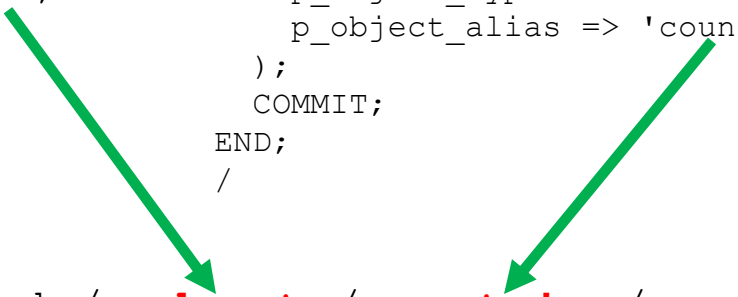
URI: http://144.21.70.32:8080/ords/code1/

AUTOREST for a schema

```
p_auto_rest_auth => TRUE,
```

```
BEGIN
  ORDS.ENABLE_SCHEMA(
      p_enabled             => TRUE,
      p_schema              => 'OOW18AUTO',
      p_url_mapping_type    => 'BASE_PATH',
      p_url_mapping_pattern => 'codeauto',
      p_auto_rest_auth      => TRUE);

  COMMIT;
END;
/
```

```
BEGIN
 ORDS.ENABLE_OBJECT(
    p_enabled      => TRUE,
    p_schema       => 'OOW18AUTO',
    p_object        => 'COUNTRIES',
    p_object_type  => 'TABLE',
    p_object_alias => 'countries'
  );
  COMMIT;
END;
/
```



http://144.21.70.32:8080/ords/**codeauto**/**countries**/

AUTOREST for a schema - metadata

http://144.21.70.32:8080/ords/codeauto/**metadata-catalog**/countries/

```
name     "COUNTRIES"
primarykey        0        "country_id"
members           0        name     "country_id"              type     "NUMBER"
                  1        name     "country_iso_code"        type     "CHAR"
                  2        name     "country_name"            type     "VARCHAR2"

links    0        rel      "collection"
href     http://144.21.70.32:8080/ords/codeauto/metadata-catalog/

mediaType         "application/json"
         1        rel      "canonical"
href     http://144.21.70.32:8080/ords/codeauto/metadata-catalog/countries/
.
.
.
```

AUTOREST for a schema - queries

`http://144.21.70.32:8080/ords/codeauto/countries/?q={"country_iso_code":"US"}`

{"country_id":52790,"country_iso_code":"US","country_name":"United States of America".....}


```
country_id:        52790
country_iso_code:  "US"
country_name:      "United States of America"
links:
  0:
    rel:           "self"
    href:          "http://144.21.70.32:8080/ords/codeauto/countries/52790"
  1:
```


AUTOREST for a schema - insert

```
curl -X POST -H "Content-Type: application/json" \
-d '{"country_id":1,"country_iso_code":"UR","country_name":"Uruguay"}' \
http://144.21.70.32:8080/ords/codeauto/countries/
```

```
select * from countries order by country_id;

COUNTRY_ID CO COUNTRY_NAME
---------- -- ----------------------------------------
         1 UR Uruguay
     52769 SG Singapore
     52770 IT Italy
     52771 CN China
.
.
```

AUTOREST for a schema - delete

curl -i -X DELETE
http://144.21.70.32:8080/ords/codeauto/countries/?q=%7B%22country_iso_code%22:%22URY%22%7D

HTTP/1.1 **200** OK
Date: Thu, 18 Oct 2018 10:22:42 GMT
Content-Type: application/json
Transfer-Encoding: chunked

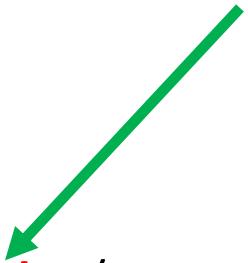
```
select * from countries where country_id=1;

no rows selected
```

Also, that was all real SLOW! (it's getting better!)

Think carefully before using AUTOREST!

Enable REST for a schema

```
BEGIN
    ORDS.ENABLE_SCHEMA(
        p_enabled             => TRUE,
        p_schema              => 'OOW18',
        p_url_mapping_type    => 'BASE_PATH',
        p_auto_rest_auth      => FALSE,
        p_url_mapping_pattern => 'code1');
    commit;
END;
```

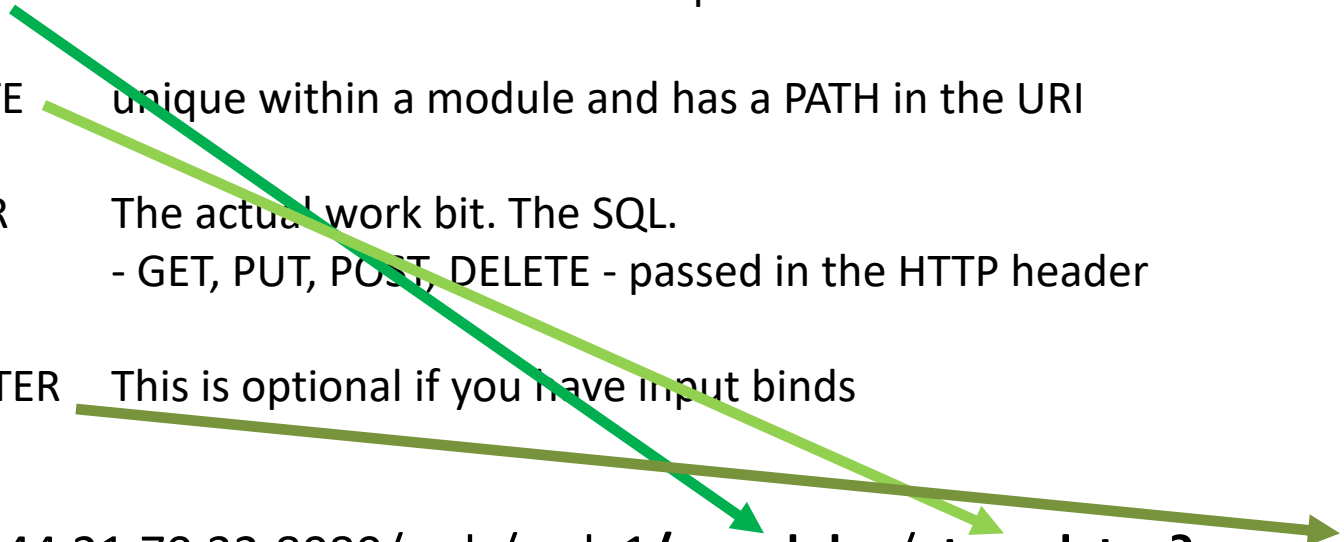


URI: http://144.21.70.32:8080/ords/code1/

Enable REST for a schema

Now we need to create

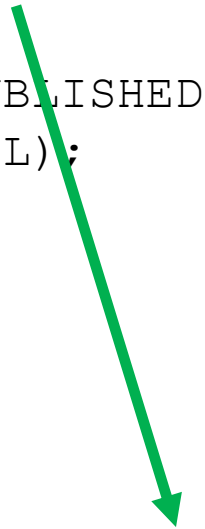
- a MODULE — a container for 1 or more Templates and has a PATH in the URI
- a TEMPLATE — unique within a module and has a PATH in the URI
- a HANDLER — The actual work bit. The SQL.
  - GET, PUT, POST, DELETE - passed in the HTTP header
- a PARAMETER — This is optional if you have input binds



URI: http://144.21.70.32:8080/ords/code1**/<module>/<template>?<parameter>=**

Enable REST : Module

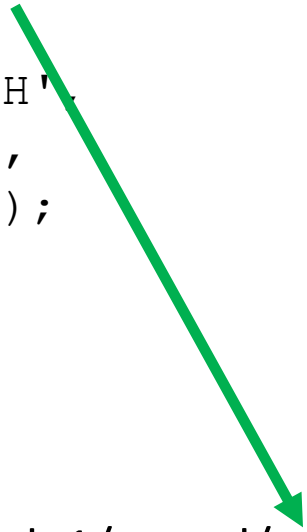
```
ORDS.DEFINE_MODULE(
      p_module_name     => 'neural',
      p_base_path       => '/neural/',
      p_items_per_page  =>  0,
      p_status          => 'PUBLISHED',
      p_comments        => NULL);
```



URI: http://144.21.70.32:8080/ords/code1/neural/<template>

Enable REST : Template

```
ORDS.DEFINE_TEMPLATE(
      p_module_name     => 'neural',
      p_pattern         => 'nw1',
      p_priority        => 0,
      p_etag_type       => 'HASH',
      p_etag_query      => NULL,
      p_comments        => NULL);
```



URI: http://144.21.70.32:8080/ords/code1/neural/**nw1**

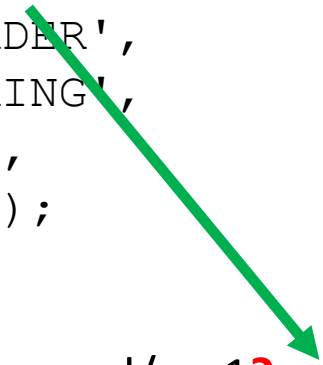
Enable REST : Handler

```
ORDS.DEFINE_HANDLER(
     p_module_name    => 'neural',
     p_pattern        => 'nw1',
     p_method         => 'GET',
     p_source_type    => 'json/collection',
     p_items_per_page =>  5,
     p_mimes_allowed  => '',
     p_comments       => NULL,
     p_source         =>
'select country_name,country_iso_code from countries
 where country_iso_code like :code order by country_name'
      );
```

URI: http://144.21.70.32:8080/ords/code1/neural/**nw1**

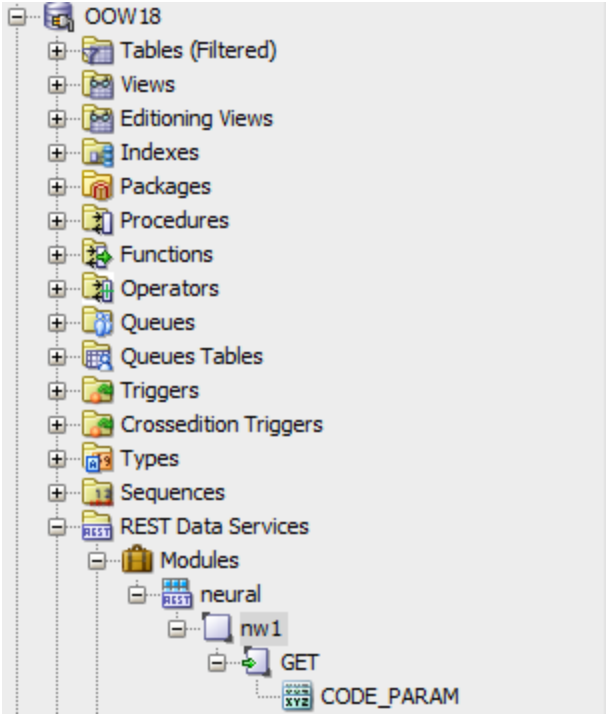
Enable REST : Handler

```
ORDS.DEFINE_PARAMETER(
     p_module_name         => 'neural',
     p_pattern             => 'nw1',
     p_method              => 'GET',
     p_name                => 'CODE_PARAM',
     p_bind_variable_name  => 'code',
     p_source_type         => 'HEADER',
     p_param_type          => 'STRING',
     p_access_method       => 'IN',
     p_comments            => NULL);
```



URI: http://144.21.70.32:8080/ords/code1/neural/nw1?code=

Do it all in SQL Developer



URI: http://144.21.70.32:8080/ords/code1/neural/nw1?code=

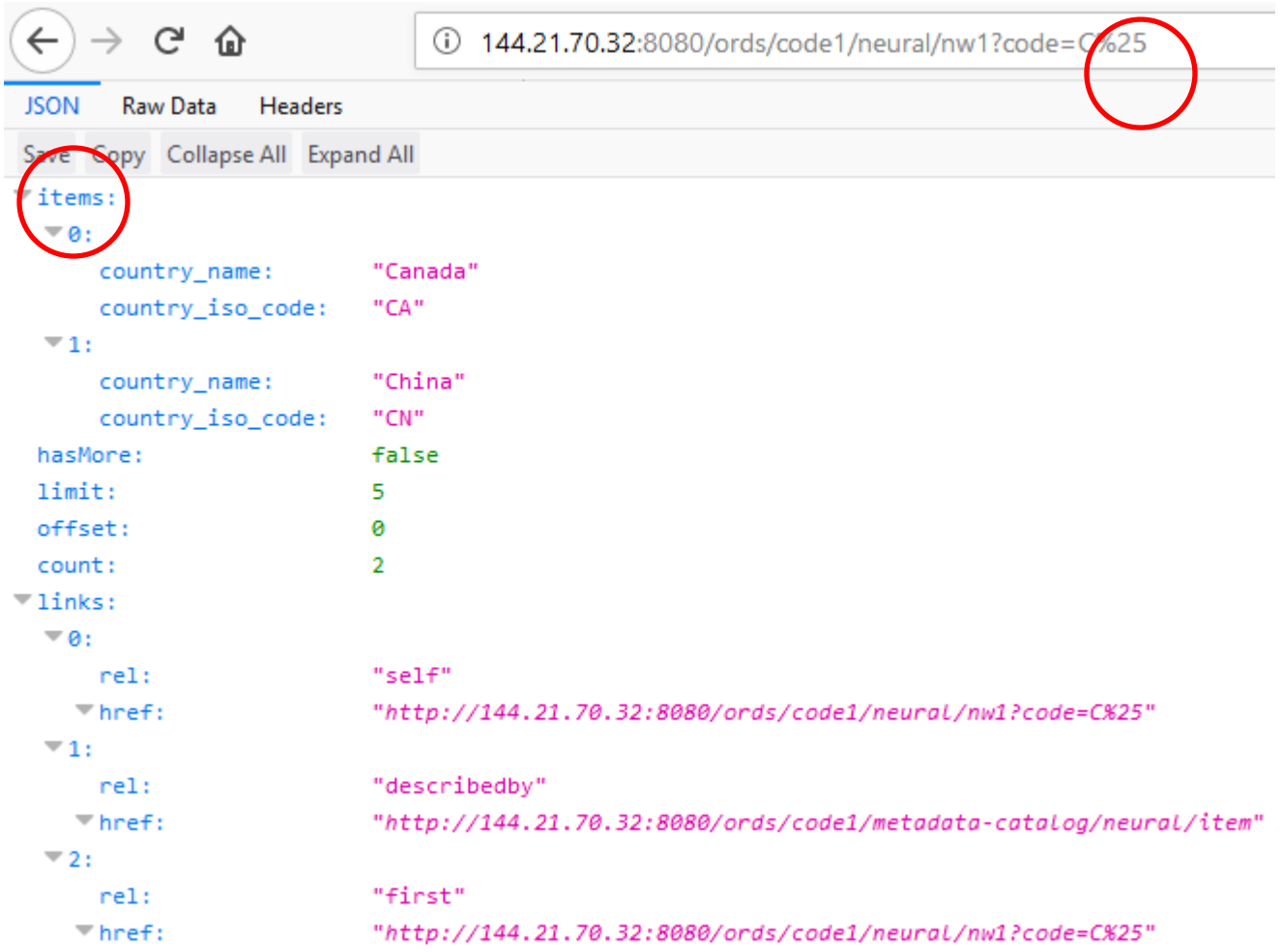
Use the URI

http://144.21.70.32:8080/ords/code1/neural/nw1**?code=**

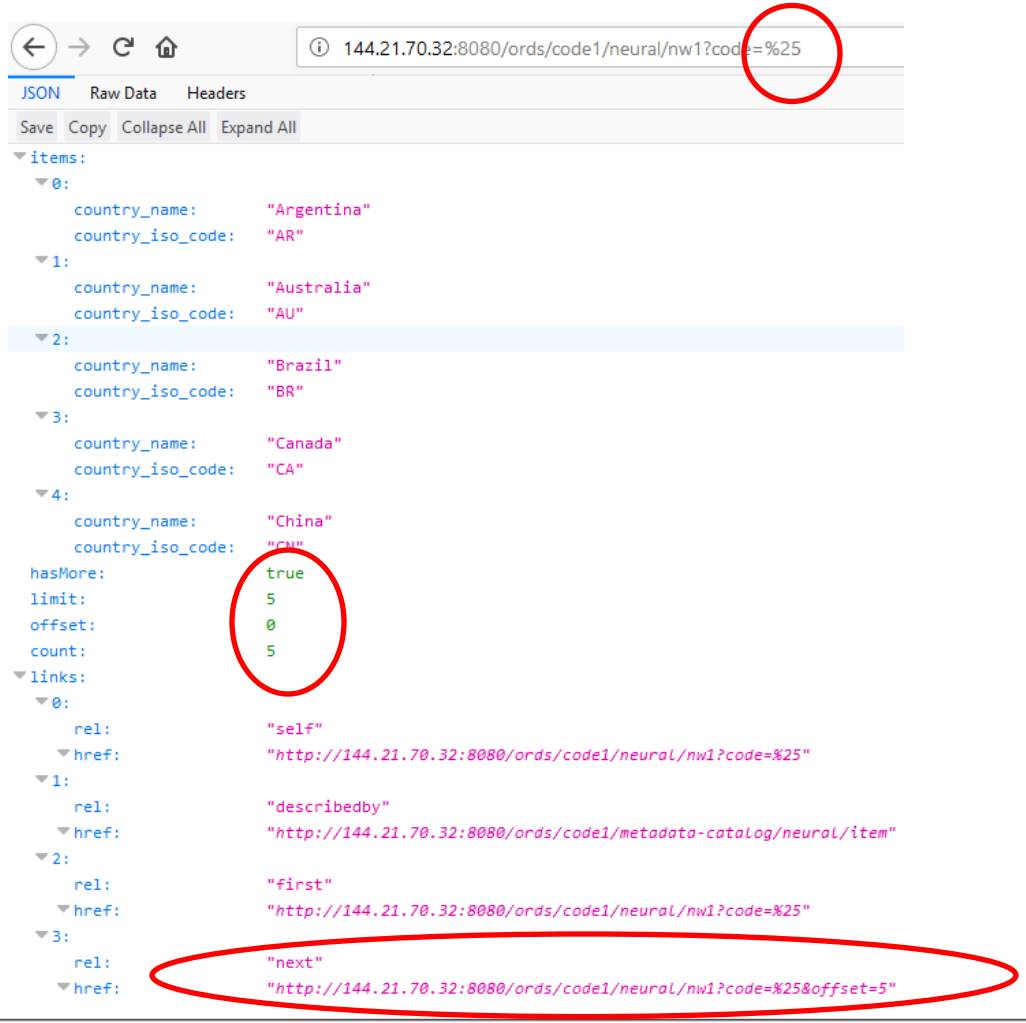
http://144.21.70.32:8080/ords/code1/neural/nw1**?code=C%25**


```
{"items":[{"country_name":"Canada","country_iso_code":"CA"},{"countr
y_name":"China","country_iso_code":"CN"}],"hasMore":false,"limit":5,
"offset":0,"count":2,"links":[{"rel":"self","href":"http://144.21.70
.32:8080/ords/code1/neural/nw1?code=C%25"},{"rel":"describedby","hre
f":"http://144.21.70.32:8080/ords/code1/metadata-
catalog/neural/item"},{"rel":"first","href":"http://144.21.70.32:808
0/ords/code1/neural/nw1?code=C%25"}]}
```

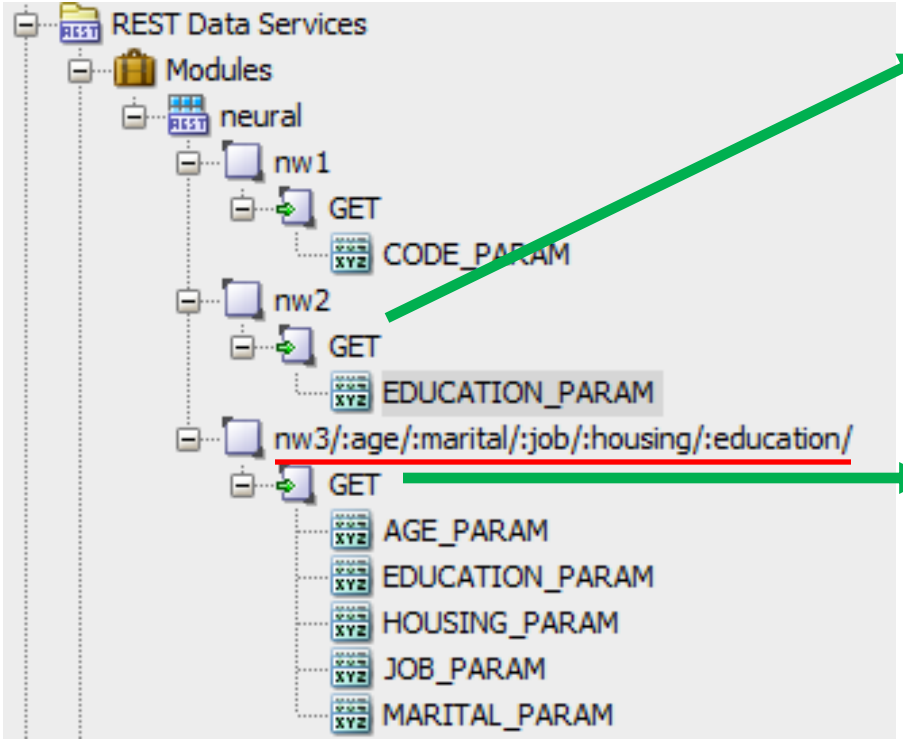

Use the URI


144.21.70.32:8080/ords/code1/neural/nw1?code=C%25
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
items:
0:
country_name: "Canada"
country_iso_code: "CA"
1:
country_name: "China"
country_iso_code: "CN"
hasMore: false
limit: 5
offset: 0
count: 2
links:
0:
rel: "self"
href: "http://144.21.70.32:8080/ords/code1/neural/nw1?code=C%25"
1:
rel: "describedby"
href: "http://144.21.70.32:8080/ords/code1/metadata-catalog/neural/item"
2:
rel: "first"
href: "http://144.21.70.32:8080/ords/code1/neural/nw1?code=C%25"

Use the URI


144.21.70.32:8080/ords/code1/neural/nw1?code=%25
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
items:
0:
country_name: "Argentina"
country_iso_code: "AR"
1:
country_name: "Australia"
country_iso_code: "AU"
2:
country_name: "Brazil"
country_iso_code: "BR"
3:
country_name: "Canada"
country_iso_code: "CA"
4:
country_name: "China"
country_iso_code: "CN"
hasMore: true
limit: 5
offset: 0
count: 5
links:
0:
rel: "self"
href: "http://144.21.70.32:8080/ords/code1/neural/nw1?code=%25"
1:
rel: "describedby"
href: "http://144.21.70.32:8080/ords/code1/metadata-catalog/neural/item"
2:
rel: "first"
href: "http://144.21.70.32:8080/ords/code1/neural/nw1?code=%25"
3:
rel: "next"
href: "http://144.21.70.32:8080/ords/code1/neural/nw1?code=%25&offset=5"

And now with the ML code!



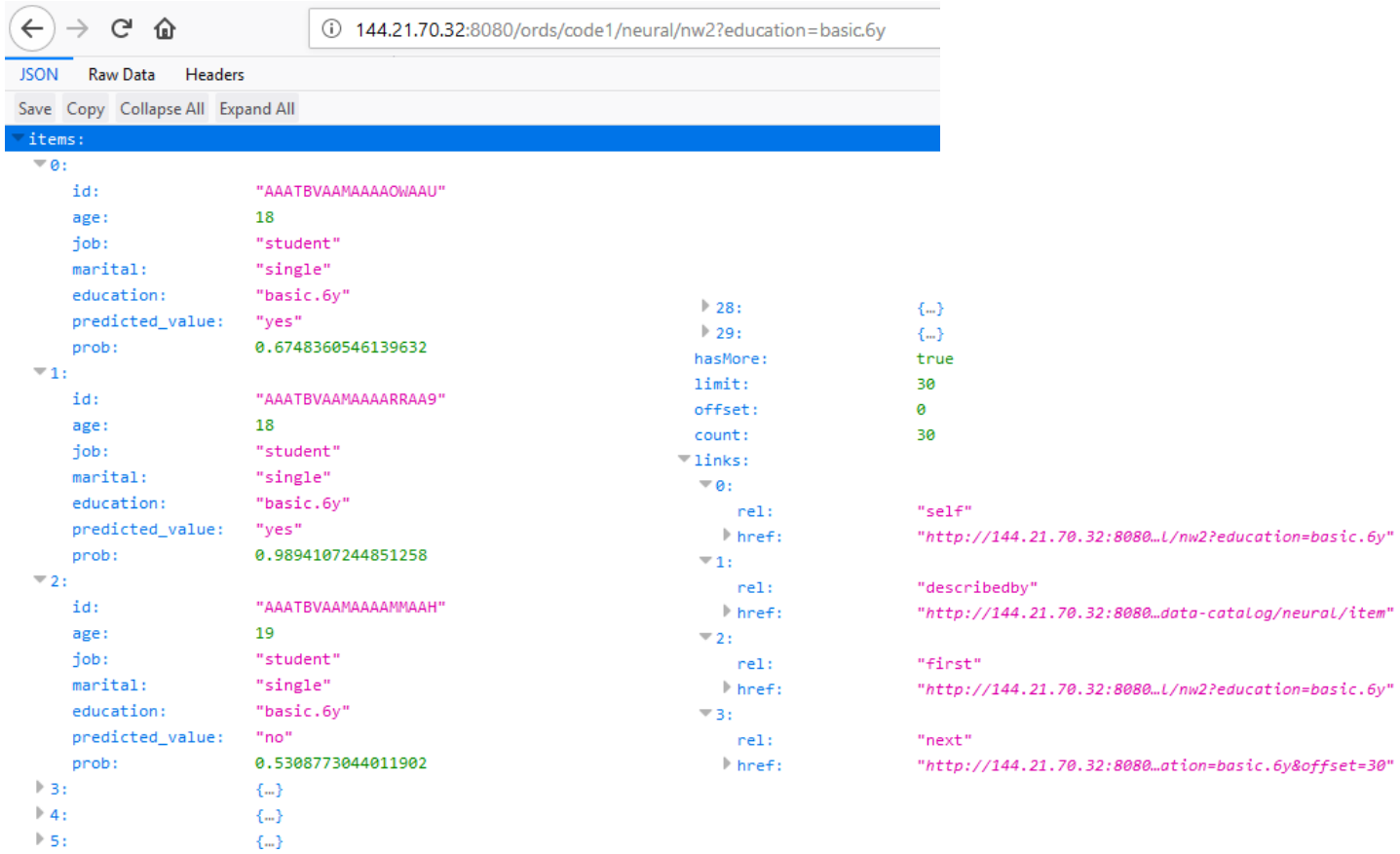

```
SELECT id, age,job, marital, education,
       PREDICTION(DEMO_NNET_MODEL_3 USING *) Predicted_Value,
       PREDICTION_PROBABILITY(DEMO_NNET_MODEL_3 USING *) Prob
FROM banking_marketing_data
where education like NVL(:education,'%')
order by age

SELECT prediction(DEMO_NNET_MODEL_3
          USING :age AS age,
            :marital AS marital,
                :job AS job,
            :housing AS housing,
           :education AS education)  Predicted_Value,
        prediction_probability(DEMO_NNET_MODEL_3, 'no'
          USING :age AS age,
            :marital AS marital,
                :job AS job,
            :housing AS housing,
           :education AS education) Predicted_Prob_NO,
        prediction_probability(DEMO_NNET_MODEL_3, 'yes'
          USING :age AS age,
            :marital AS marital,
                :job AS job,
            :housing AS housing,
           :education AS education) Predicted_Prob_YES
FROM dual
```

And now with the ML code!

```
 ORDS.DEFINE_TEMPLATE(
      p_module_name    => 'neural',
      p_pattern        => 'nw2',
      p_priority       => 1,
      p_etag_type      => 'HASH',
      p_etag_query     => NULL,
      p_comments       => NULL);
  ORDS.DEFINE_HANDLER(
      p_module_name    => 'neural',
      p_pattern        => 'nw2',
      p_method         => 'GET',
      p_source_type    => 'json/collection',
      p_items_per_page =>  30,
      p_mimes_allowed  => '',
      p_comments       => NULL,
      p_source         =>
'SELECT id, age,job, marital, education,
       PREDICTION(DEMO_NNET_MODEL_3 USING *) Predicted_Value,
       PREDICTION_PROBABILITY(DEMO_NNET_MODEL_3 USING *) Prob
FROM banking_marketing_data
where education like NVL(:education,''%'')
order by age'
      );
  ORDS.DEFINE_PARAMETER(
      p_module_name        => 'neural',
      p_pattern            => 'nw2',
      p_method             => 'GET',
      p_name               => 'EDUCATION_PARAM',
      p_bind_variable_name => 'education',
      p_source_type        => 'HEADER',
      p_param_type         => 'STRING',
      p_access_method      => 'IN',
      p_comments           => NULL);
```

http://144.21.70.32:8080/ords/code1/neural/nw2?education=basic.6y

And now with the ML code!

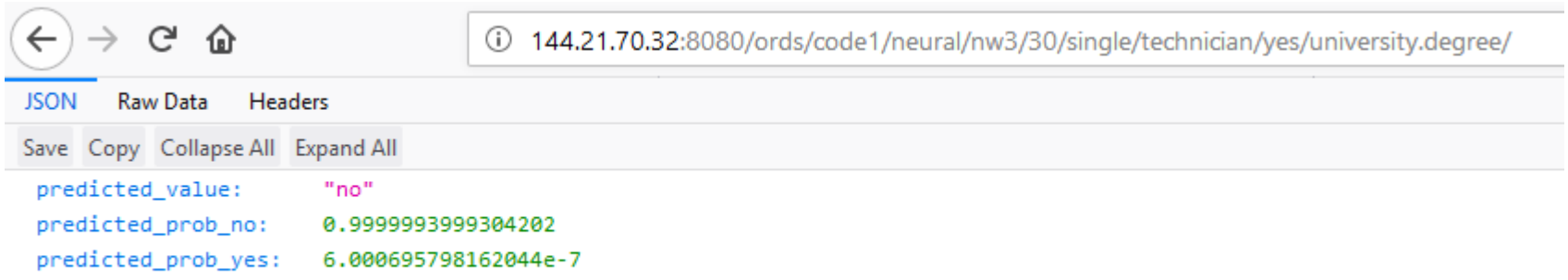


And now with the ML code!

http://144.21.70.32:8080/ords/code1/neural/nw3/30/single/technician/yes/university.degree/

```
 ORDS.DEFINE_TEMPLATE(
      p_module_name    => 'neural',
      p_pattern        => 'nw3/:age/:marital/:job/:housing/:education/',
      p_priority       => 0,
      p_etag_type      => 'HASH',
      p_etag_query     => NULL,
      p_comments       => NULL);
  ORDS.DEFINE_HANDLER(
      p_module_name    => 'neural',
      p_pattern        => 'nw3/:age/:marital/:job/:housing/:education/',
      p_method         => 'GET',
      p_source_type    => 'json/query;type=single',
      p_items_per_page =>  0,
      p_mimes_allowed  => '',
      p_comments       => NULL,
      p_source         =>
'SELECT prediction(DEMO_NNET_MODEL_3
          USING :age AS age,
            :marital AS marital,
                :job AS job,
            :housing AS housing,
           :education AS education)  Predicted_Value,
        prediction_probability(DEMO_NNET_MODEL_3, ''no''
          USING :age AS age,
            :marital AS marital,
                :job AS job,
            :housing AS housing,
           :education AS education) Predicted_Prob_NO,
        prediction_probability(DEMO_NNET_MODEL_3, ''yes''
          USING :age AS age,
            :marital AS marital,
                :job AS job,
            :housing AS housing,
           :education AS education) Predicted_Prob_YES
FROM dual');
```

And now with the ML code!



Python

Python 3.6
Add "Requests" to Python - a Python HTTP library that simplifies working with HTTP (https://github.com/requests/requests/tarball/master) and install using pip

```
C:\Users\neil\requests-requests-a6cd380>pip install .
Processing c:\users\neil_\onedrive\desktop\ords\requests-requests-
a6cd380
Collecting chardet<3.1.0,>=3.0.2 (from requests==2.19.1)

sted.org/packages/bc/a9/01ffebfb562e4274b6487b
4c51f14098443b8/chardet-3.0.4-py2.py3-none-

| 143kB 499kB/s
2.5 (from requests==2.19.1)
  Downloading
https://files.pythonhosted.org/packages/4b/2a/0276479a4b3caeb8a8c1af
2f8e4355746a97fab05a372e4a2c6a6b876165/idna-2.7-py2.py3-none-any.whl
(58kB)
    100% |                                | 61kB 627kB/s
Collecting urllib3<1.24,>=1.21.1 (from requests==2.19.1)
  Downloading
https://files.pythonhosted.org/packages/bd/c9/6fdd990019071a4a32a5e7
cb78a1d92c53851ef4f56f62a3486e6a7d8ffb/urllib3-1.23-py2.py3-none-
any.whl (133kB)
    100% |                                | 143kB 941kB/s
Collecting certifi>=2017.4.17 (from requests==2.19.1)
  Downloading
https://files.pythonhosted.org/packages/df/f7/04fee6ac349e915b82171f
8e23cee63644d83663b34c539f7a09aed18f9e/certifi-2018.8.24-py2.py3-
none-any.whl (147kB)
    100% |                                | 153kB 891kB/s
Installing collected packages: chardet, idna, urllib3, certifi,
requests
Running setup.py install for requests ... done
Successfully installed certifi-2018.8.24 chardet-3.0.4 idna-2.7
requests-2.19.1 urllib3-1.23
```

Python - ML3 (single row return)

```
import json
import requests
resp =
requests.get('http://144.21.70.32:8080/ords/code1/neural/nw3/30/single/technician/yes/university.degree/')
json_data = resp.json()
print (json.dumps(json_data, indent=2))

resp =
requests.get('http://144.21.70.32:8080/ords/code1/neural/nw3/30/married/technician/yes/university.degree/')
json_data = resp.json()
print (json.dumps(json_data, indent=2))
```

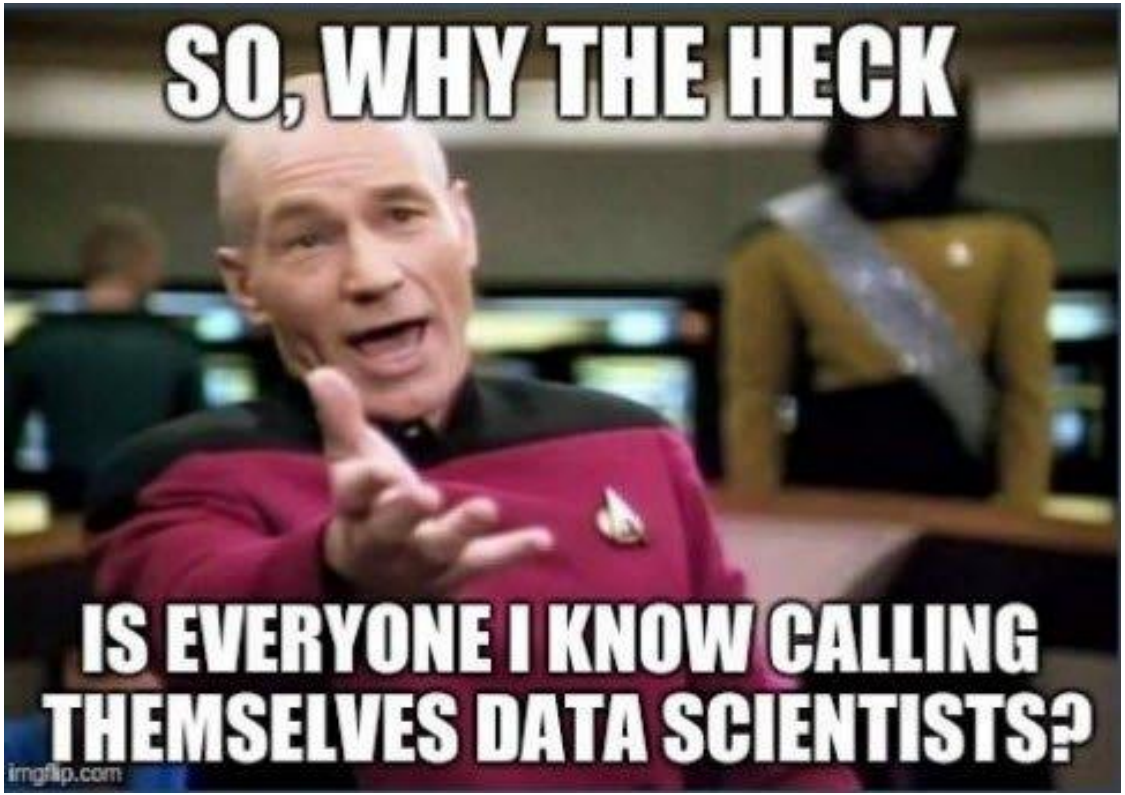

```
{
  "predicted_value": "no",
  "predicted_prob_no": 0.9999993999304202,
  "predicted_prob_yes": 6.000695798162044e-07
}
{
  "predicted_value": "no",
  "predicted_prob_no": 0.9999950557147201,
  "predicted_prob_yes": 4.944285279825768e-06
}
```

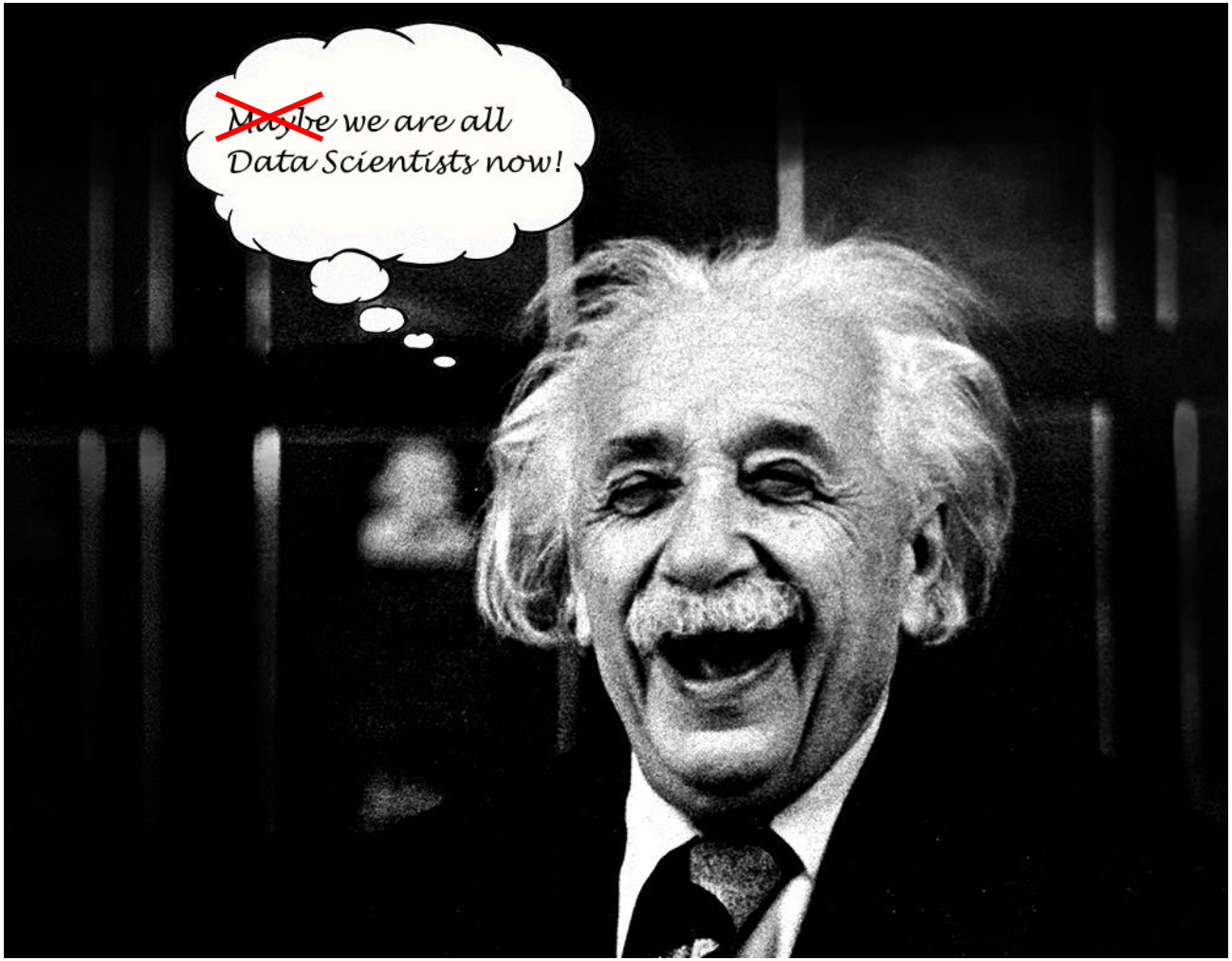

Python - ML2 (multi-row return)

```
import json
import requests
resp = requests.get('http://144.21.70.32:8080/ords/code1/neural/nw2?education=basic.6y')
for item in resp.json()["items"]:
  l_age = item["age"]
  l_job = item["job"]
  l_mar = item["marital"]
  l_edu = item["education"]
  l_pre = item["predicted_value"]
  l_pro = item["prob"]
  print (l_age, l_job, l_mar, l_edu, l_pre, l_pro)
```

```
18 student single basic.6y yes 0.6748360546139632
18 student single basic.6y yes 0.9894107244851258
19 student single basic.6y no 0.5308773044011902
19 student single basic.6y yes 0.9230338821636889
19 student single basic.6y yes 0.9993723454663064
19 student single basic.6y yes 0.999595325304911
22 blue-collar single basic.6y no 0.9999999999848361
22 blue-collar single basic.6y yes 0.9380215286535086
22 blue-collar single basic.6y no 0.9999999998548642
.
```

Summary


SO, WHY THE HECK
IS EVERYONE I KNOW CALLING THEMSELVES DATA SCIENTISTS?
imgflip.com


~~Maybe~~ we are all
Data Scientists now!

ML 101: Oracle Data Miner/Oracle Autonomous Data Warehouse Cloud Machine Learning Notebook - BYOL [HOL5759]

SPEAKERS

Brendan Tierney, ., Oralytics

Charlie Berger, Sr. Director Product Management, Machine Learning, AI and Cognitive Analytics, Oracle

Derrick Cameron, Oracle

Code One Tracks: Database, Big Data, and Data Science

Session Type: HOL (Hands-on Lab) Session

Monday, Oct 22, 02:45 PM - 04:45 PM | Moscone West - Overlook 2A (HOL)

Brendan Tierney

Oracle Developer Champion

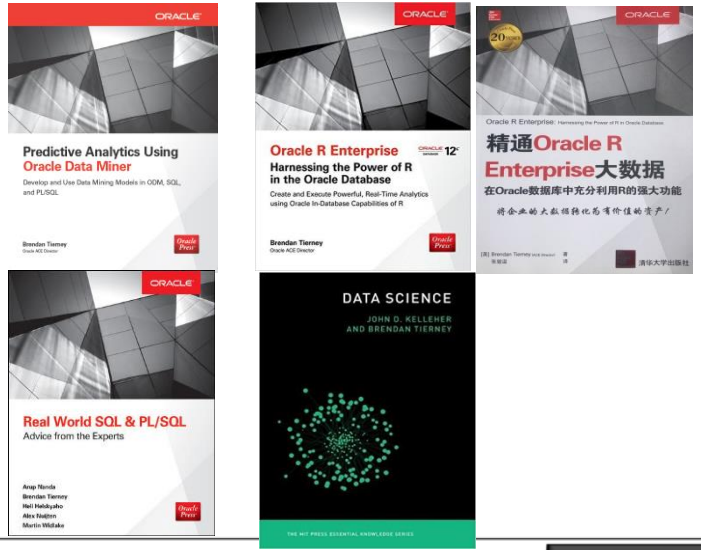
Oracle ACE Director

Oracle Groundbreaker Ambassador

e. brendan.tierney@oralytics.com

t. @brendantierney

w. www.oralytics.com

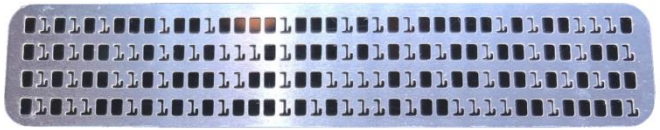


Neil Chandler

Oracle ACE Director

t. @chandlerDBA

w. http://chandlerDBA.com



APPS18 JDE18 TECH18

3-5 DEC 2018 #ukoug_apps18 | 4-5 DEC 2018 #ukoug_jde18 | 3-5 DEC 2018 #ukoug_tech18

Registration is now open for UKOUG's co-located conferences at the ACC, Liverpool

View the agendas & register now

apps18.ukoug.org

TOPICS	KEYNOTES
Analytics Apps Tech Business & Strategy Cloud E-Business Suite ERP HCM PeopleSoft	Cliff Godwin - E-Business Suite Mark Newton - ERP Cloud Sarah Henry - HCM Cloud David Bain - Peoplesoft

jde18.ukoug.org

TOPICS	KEYNOTE
Customer Stories Business Function Business Process Future	Lyle Ekdahl - SVP & General Manager, JD Edwards

tech18.ukoug.org

TOPICS	KEYNOTES
Analytics APEX Database Development Platform & DevOps Systems	John Hagerty - Analytics & Big Data Joel Kallman - APEX Dominic Giles & Maria Colgan - Database

OVERALL KEYNOTE - Richard Petley & John Abel

Oracle Groundbreaker Ambassador

Oracle Groundbreakers

Experts in Modern Development

- Cloud
- Microservices and Containers
- Java, JavaScript/Node.js, PHP, Python
- DevOps
- Continuous Delivery
- Open Source Technologies
- SQL/NoSQL Databases
- Machine Learning, AI, Chatbots

developer.oracle.com/ambassador

@groundbreakers