Module 4: Predictive/Ensemble Analytic Functions

Teradata Vantage Analytics Workshop
BASIC

Objectives

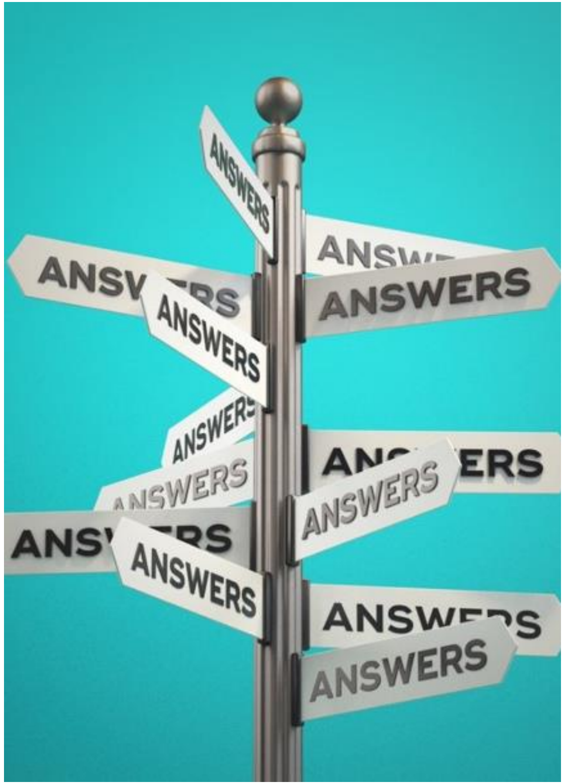
After completing this module, you will be able to:

- Write queries using these Teradata Vantage predictive and other analytic functions:
  - **Correlation**
  - **Decision Forest Trees** (with **Confusion Matrix**)
  - **XGBoost**

For more info go to docs.teradata.com click Teradata Vantage, download: Teradata Vantage Machine Learning Engine Analytic Function Reference guide

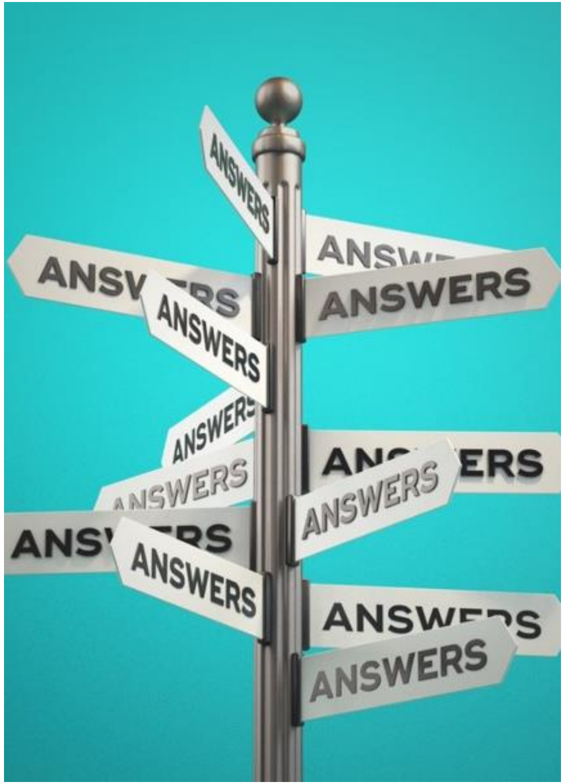
Topics

- Predictive Analytics Overview
- Correlation
- Ensemble Functions
  - Decision Forest Trees (with ConfusionMatrix)
  - XGBoost
- Summary



Current Topic – Predictive Analytics Overview

- **Predictive Analytics Overview**
- Correlation
- Ensemble Functions
  - Decision Forest Trees (with ConfusionMatrix)
  - XGBoost
- Summary



Model and Scoring: Definitions

Model - A machine learning model can be a mathematical representation of a real-world process. The learning algorithm finds patterns in the training data such that the input parameters correspond to the target. The output of the training process is a machine learning model which you can then use to make predictions.

Scoring - (also called Prediction) is the process of generating values based on a trained machine learning model, given some new input data. The values or scores that are created can represent predictions of future values, but they might also represent a likely category or outcome.

In real-world examples, this is typically a 2-Step Process:

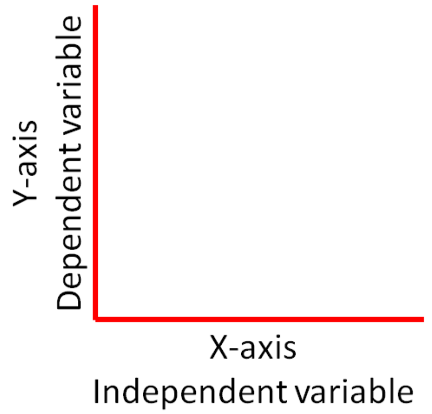
1. First create the **Model** on the TRAIN (Known data) and
2. Then **Score** (Make Prediction) on the PRODUCTION (Unknown) data

Optionally may also view Accuracy of the Model if the Known data is available (using either Confusion Matrix function) or simple calculations

Independent and Dependent Variables: Definitions

Independent Variable - Variable you have control over, what you can choose and manipulate. It is usually what you think will affect the Dependent variable.

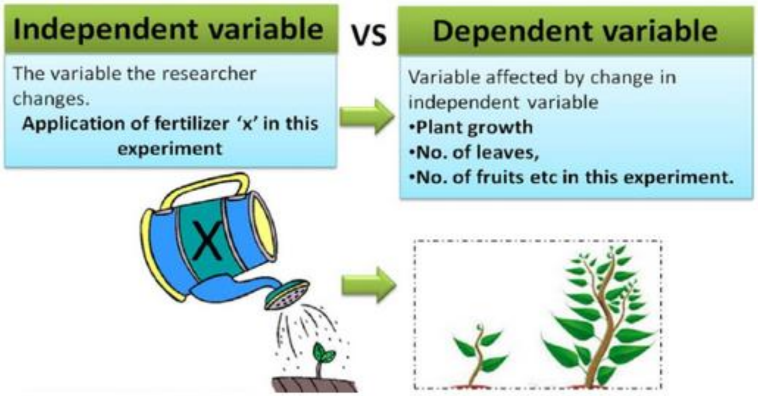
Dependent Variable - What you measure in the experiment and what is affected during the experiment. The Dependent variable responds to the independent variable. It is called Dependent because it depends on Independent variable. In a scientific experiment, you cannot have a Dependent variable without an Independent variable.



Independent variables are sometimes called Predictors or X-variables

Dependent variables are sometimes called the Response, Target or Y-variable

For remainder of this Module, we'll use X-var, Y-var terms



Independent Variable Types: Definitions

Numeric - This variable has a Numeric data type and can mathematically be aggregated. Numeric variables are classified into:

1. **Continuous** - Value obtained by Measuring. Infinity possible values
   Examples: Height, Weight, Distance, Time
2. **Discrete** - Value obtained by Counting. Have specific values
   Examples: # of students, # of books

Categorical - Can be Numeric or String data type. However arithmetic operations cannot be performed on the values. Categorical variables are classified into:

1. **Nominal** - Label or Name:
   Examples: Eye Color, Country, Marital Status (Cannot order these)
2. **Ordinal** - Class Position:
   Examples: Pollution = Low, Medium, High (Can order these)
   Airline seat = First, Second, Third (Can order these)

Most Predictive Functions Operate in 2-Step Process

1. **Known Data** (Split TRAIN 80% / TEST 20%) –Typically consists of X-variables and Y-variable. Run data through algorithm and create Model to use on Unknown data
2. **Unknown data** (Production set) – Consists of only X-variables. You run the Unknown data through function using the Model to predict Y-variable

	id	year	color	type	origin	stolen
		x	x	x	x	y
1	1	1	Red	Sports	Domestic	Yes
2	3	2	Red	Sports	Domestic	Yes
3	5	3	Yellow	Sports	Imported	Yes
4	7	4	Yellow	SUV	Imported	Yes
5	9	12	Red	SUV	Imported	No
6	2	8	Red	Sports	Domestic	No
7	4	9	Yellow	Sports	Domestic	No
8	6	10	Yellow	SUV	Imported	No
9	8	11	Yellow	SUV	Domestic	No
10	10	5	Red	Sports	Imported	Yes

Known Data - Train

Build Model (Train Set)

	id	year	color	type	origin	stolen
		x	x	x	x	y
1	21	1	Red	Sports	Domestic	Yes
2	43	2	Red	Sports	Domestic	Yes
3	65	3	Yellow	Sports	Imported	Yes

Known Data - Test

Evaluate Model (Test Set)

	id	year	color	type	origin
		x	x	x	x
1	11	1	Red	SUV	Domestic
2	13	3	Red	SUV	Domestic
3	15	5	Red	SUV	Domestic
4	17	7	Red	SUV	Domestic
5	12	2	Red	SUV	Domestic
6	14	4	Red	SUV	Domestic
7	16	6	Red	SUV	Domestic

Unknown Data - Prod

Deploy Model (Production Data)

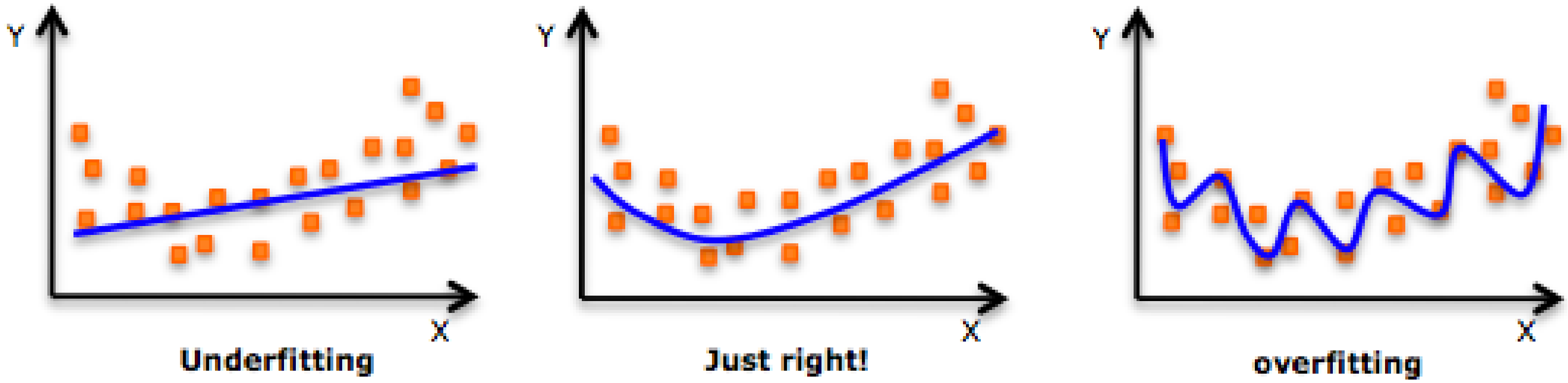
Prediction of car being Stolen (Y-var)

	test_id	prediction
1	11	Yes
2	12	Yes
3	13	Yes
4	14	Yes
5	15	Yes
6	16	Yes
7	17	No

Answer Set

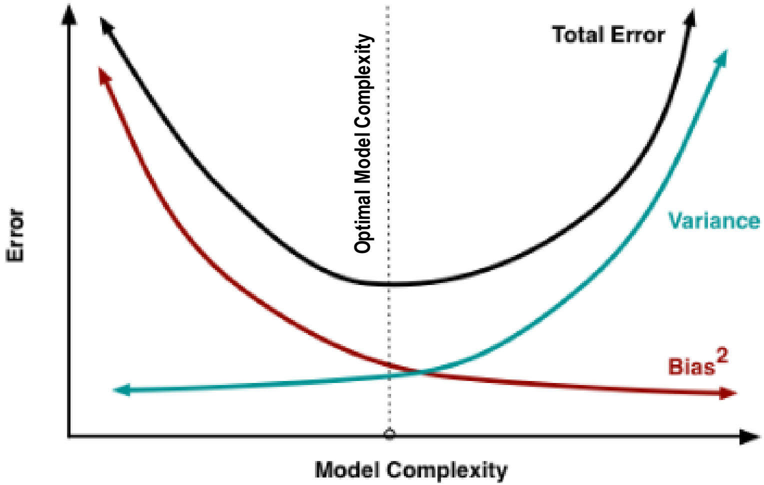
Model Overfitting and Underfitting

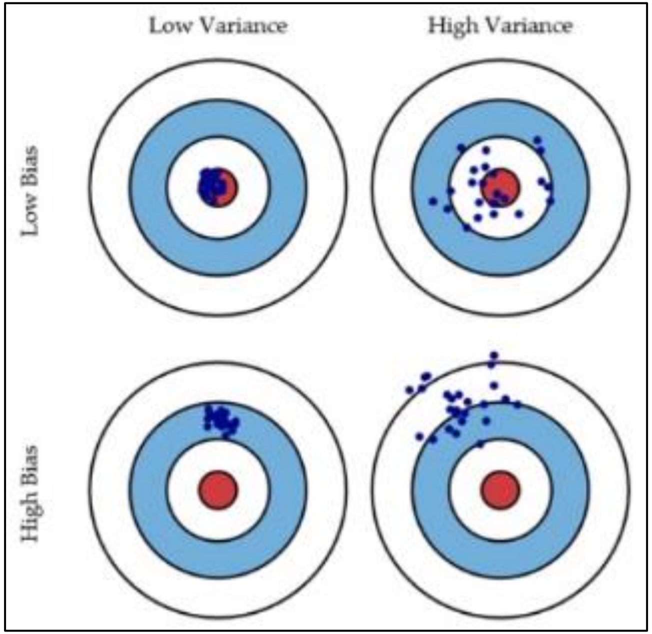
- Goal of Machine Learning is to Model the Pattern and ignore the Noise. Anytime an algorithm is fitting the Noise in addition to Pattern, it is Overfitting.
- In middle diagram, algorithm returns the best fit line given those points while the Right diagram is Modeling the Noise. As can be seen from this example, a way to reduce Overfitting is then to artificially penalize the Noise.



Model Building: Bias and Variance

- Goal is to balance Low Bias and Low Variance
- To overcome High Bias (Underfitting), add more X-variables to reduce Bias
- As add more X-variables, complexity increases which results in increasing Variance, less Bias (Overfitting)





High 'Coefficients' means these X-variables more important in predicting Y-variable
https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/l2-regularization

What is Supervised Learning?

- Supervised learning is where you have Input variables (X) and an Output variable (Y), and you use an algorithm to learn the mapping function from the input to the output
- It is called Supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers, the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance
- Supervised learning can be grouped into:
  - **Classification**: A Classification problem is when the Output variable(Y) is a category, such as 'red' or 'blue' or 'disease' and 'no disease'
  - **Regression**: A Regression problem is when the Output variable(Y) is a real value, such as 'dollars' or 'weight'

Supervised Learning Example

Supervised Learning – **Training** set composed of X-variables and Y-variables.

The Y-variable is labeled as either Classification or Regression (value)

	x	x	x	x	y
id	year	color	type	origin	stolen
1	1	Red	Sports	Domestic	Yes
2	8	Red	Sports	Domestic	No
3	2	Red	Sports	Domestic	Yes
4	9	Yellow	Sports	Domestic	No
5	3	Yellow	Sports	Imported	Yes

	x	x	x	x	y
custid	age	gender	marital	ccnum	profit
1	36	M	M	1	10
2	32	M	S	1	-30
3	38	M	M	2	35
4	40	M	S	1	-45
5	44	M	M	2	25

		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	y
	car_name	sports	suv	wagon	minivan	pickup	awd	rwd	engine	cylinders	horsepower	city_mpg	highway_mpg	weight	wheelbase	length	width	retail_price	dealer_price	price_class
1	Acura 3.5 RL Navigation	0	0	0	0	0	0	0	3,500,000	6	225	18	24	3,893	115	197	72	46,100	41,10	5
2	Acura TL	0	0	0	0	0	0	0	3,200,000	6	270	20	28	3,575	108	186	72	33,195	30,29	3
3	Audi A4 1.8T convertible	0	0	0	0	0	0	0	1,800,000	4	170	23	30	3,638	105	180	70	35,940	32,50	4
4	Audi A4 3.0 convertible	0	0	0	0	0	0	0	3,000,000	6	220	20	27	3,814	105	180	70	42,490	38,32	4
5	Audi A4 3.0 Quattro convertible	0	0	0	0	0	1	0	3,000,000	6	220	18	25	4,013	105	180	70	44,240	40,07	4

Examples of Supervised ML: Naïve Bayes, Decision Trees, GLM, LARS, XGBoost

What is Unsupervised Learning?

Unsupervised Learning – Algorithm is not provided with information about the Y-variable. The machine is tasked with discovering this.

Here's a dataset of cereals. We ask the algorithm to discover how many 'like' clusters there are.

Answer set says 4 clusters of cereals.

name	calorie X	protein X	fat X	sodium X	fiber X	carbo X	sugar X	potass X	vitamin X
100%_Bran	70	4	1	130	10.0	5.0	6	280	25
100%_Natural_Bran	120	3	5	15	2.0	8.0	8	135	0
All-Bran_with_Extra_Fiber	50	4	0	140	14.0	8.0	0	330	25
Bran_Chex	90	2	1	200	4.0	15.0	6	125	25
Bran_Flakes	90	3	0	210	5.0	13.0	5	190	25
Cheerios	110	6	2	290	2.0	17.0	1	105	25
Cocoa_Puffs	110	1	1	180	0.0	12.0	13	55	25
Cracklin_Oat_Bran	110	3	3	140	4.0	10.0	7	160	25
Crispix	110	2	0	220	1.0	21.0	3	30	25

canopyid	calorie X	protein X	fat X	sodium X	fiber X	carbo X	sugar X	potass X	vitamin X
1	108	2	1	187	1.1087	14.8913	8	54	23
2	85	3	1	173	7.83333	9.16667	5	246	25
3	101	2	1	172	3.9	12.4333	7	143	25
4	106	2	1	88	2.125	11.875	8	95	21

Unsupervised ML Algorithms:

Canopy, KMeans, KModes

OutlierFilter, Principal Component Analysis (PCA)

Review: Predictive Analytics Overview

Assume wish to predict column 'homestyle' below. Answer the following:

- What's the **Y**-variable? — 'homestyle' column
- How many **X**-variables? — **13** Don't count '**sn**' since it's unique ID
- How many **Numeric** **X**-vars? — $\boxed{6}$ price, lotsize, bedrooms, bathrms, stories, garagepl
- How Many **Categorical** **X**-vars? — **7** driveway, recroom, fullbase, gashw, airco, prefarea, buyer_gender
- Classification or Regression? Classification since Y-variable is not numeric value or a future value

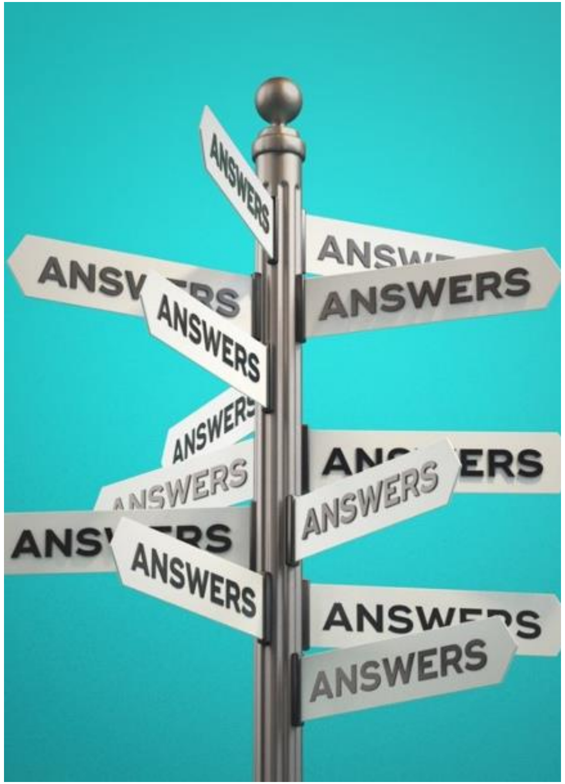
0	1	2	3	4	5	6	7	8	9	10	11	12	13	Y-var
sn	price	lotsize	bedrooms	bathrms	stories	driveway	recroom	fullbase	gashw	airco	garagepl	prefarea	buyer_gender	homestyle
91	47000	6060	3	1	1	yes	yes	yes	no	no	0	no	1	Classic
92	58000	5900	4	2	2	no	no	yes	no	no	1	no	1	Eclectic
93	163000	7420	4	1	2	yes	yes	yes	no	yes	2	no	1	bungalow
94	128000	8500	3	2	4	yes	no	no	no	yes	2	no	0	bungalow
95	123500	8050	3	1	1	yes	yes	yes	no	yes	1	no	0	bungalow
96	39000	6800	2	1	1	yes	no	no	no	no	0	no	1	Classic

If want to Predict 'Price' and had 'country' column too, now what? Normalize data, use Regression, Partition by 'country'

Current Topic – Correlation

- Predictive Analytics Overview
- **Correlation**
- Ensemble Functions
  - Decision Forest Trees (with ConfusionMatrix)
  - XGBoost

Summary



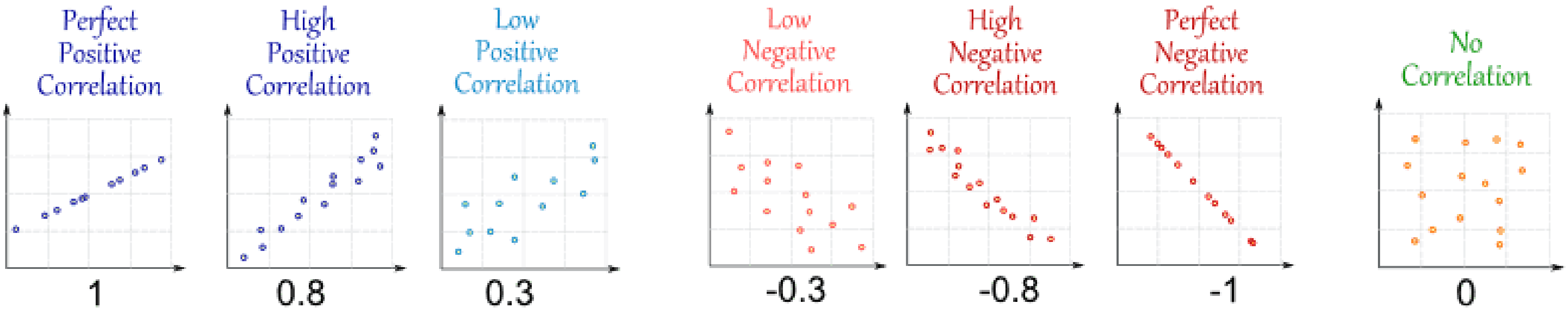
Description – Correlation

- Measuring Correlation lets you determine if the value of one variable (X-var) is useful in predicting the Y-var value. Useful when determining which X-Variables to include in your Predictive Modeling functions (NB, SVM, LinReg).
- Correlation makes no assumption as to whether 1 variable is dependent on the other. Instead, it gives degree of association between variables. It tests for interdependence of variables. Regression attempts to describe dependence of unknown variable on 1 or more known variables.

Correlation has two useful features when creating a Predictive Model

1. Which are 'best' X-variables to use when making a Prediction on Y-variable.
2. Which X-variables are highly Correlated. (MultiCollinearity occurs when have Absolute Correlation > .7 between 2 Predictors). If this case, one strategy is to remove one of X-variables from Model creation to reduce Variance in Model.

Correlation Visualized



For <u>Positive</u> correlations: as value increase in one, it increases in other

- 1 : Move in same direction
- 0 : No correlation
- -1 : Move in opposite direction

https://www.mathsisfun.com/

Correlation Is Not Causation. Correlation does not mean one thing causes the other. For example, the more firefighters you have, the more property damage you have.

http://tylervigen.com/spurious-correlations

Syntax – Correlation in VAL


```
CALL td_analyze (
'matrix',
database = input_database_name;
tablename = input_table_name;
columns = { all | column_name [,...] };
{ columnstoexclude = column_name [,...] |
groupby = column_name [,...] |
matrixoutput = { COLUMNS | VARBYTE } |
matrixtype = { COR | COV | SSCP | CSSCP | ESSCP } |
nullhanding = { IGNORE | ZERO } |
outputdatabase = output_database_name |
outputtablename = output_table_name |
overwrite = { true | false } |
where = expression
}
);
```

Correlation only works with **Numeric** data types

Arguments – Correlation

- **Columns:** Specify columns to calculate correlations. *Must be numeric*
- **GroupBy:** [Optional] Specify the names of the input columns that define the group for correlation calculation. Default behavior: All input columns belong to a single group
- **MatrixType:** COR for Correlation

Lab 1a: View the Data

Goal: Find which **X**-variables are most Correlated with '**bustout**' **Y**-variable

```
SELECT * FROM bustout_train;
```

s...	acct_no	as_of_dt_day	avg_pmt_05_mth	days_since_lstcash	max_utilization_05_mth	maxamt_epmt_v7day	times_nsf	totcash_to_line_v7day	totpmt_to_line_v7day	totpur_to_line_v7day	totpurcash_to_line_v7...	ays_since_lst_pymnt
0	817	2017-05-15	-2990.75		92.3	0	0					9
0	1491	2019-01-27	-1		183.5	0	0					78
0	605	2018-09-15	-4753.2		29.4	0	0					9
0	812	2018-05-19	-5256.6		15.23333	0	0					3
0	1052	2017-05-06	-2677		75.45	0	0					0

num_pymnt_lst_7_days	num_pymnt_lst_60_d...	pct_line_paid_lst_7_d...	pct_line_paid_lst_30_...	num_pur_lst_7_days	num_pur_lst_60_days	pct_line_pur_lst_7_days	pct_line_pur_lst_30_d...	tot_pymnt_chnl	tot_pymnt	tot_pym	_by_othr	last_12m_
0	2			2	24			0				43
0	0			0	0			0				0
0	2			5	24			0				162
0	2			7	32			0				89
0	2			8	28			0				28

num_pur_lst_7_days	num_pur_lst_60_days	pct_line_pur_lst_7_days	pct_line_pur_lst_30_d...	tot_pymnt_chnl	tot_pymnt	tot_pymnt_am	pay_by_phone	elec_pymnt	pay_in_bank	pay_by_check	pay_by_othr	last_12m_trans_ct	bustout
2	24			0			N	N	N	N	N	43	N
0	0			0			N	N	N	N	N	0	Y
5	24			0			N	N	N	N	N	162	N
7	32			0			N	N	N	N	N	89	N
8	28			0			N	N	N	N	N	28	N

- bustout_train
  - Columns
    - set_id [INTEGER, Nullable]
    - acct_no [VARCHAR (38), Nullable, PI]
    - as_of_dt_day [DATE, Nullable]
    - avg_pmt_05_mth [NUMBER (10, 2), Nullable]
    - days_since_lstcash [INTEGER, Nullable]
    - max_utilization_05_mth [NUMBER (12, 5), Nullable]
    - maxamt_epmt_v7day [NUMBER (10, 2), Nullable]
    - times_nsf [INTEGER, Nullable]
    - totcash_to_line_v7day [NUMBER (12, 5), Nullable]
    - totpmt_to_line_v7day [NUMBER (12, 5), Nullable]
    - totpur_to_line_v7day [NUMBER (12, 5), Nullable]
    - totpurcash_to_line_v7day [NUMBER (12, 5), Nullable]
    - credit_util_cur_mth [NUMBER (12, 5), Nullable]
    - credit_util_prior_5_mth [NUMBER (12, 5), Nullable]
    - credit_util_cur_to_prior_ratio [FLOAT, Nullable]
    - days_since_lst_pymnt [INTEGER, Nullable]
    - num_pymnt_lst_7_days [INTEGER, Nullable]
    - num_pymnt_lst_60_days [INTEGER, Nullable]
    - pct_line_paid_lst_7_days [NUMBER (12, 5), Nullable]
    - pct_line_paid_lst_30_days [NUMBER (12, 5), Nullable]
    - num_pur_lst_7_days [INTEGER, Nullable]
    - num_pur_lst_60_days [INTEGER, Nullable]
    - pct_line_pur_lst_7_days [NUMBER (12, 5), Nullable]
    - pct_line_pur_lst_30_days [NUMBER (12, 5), Nullable]
    - tot_pymnt_chnl [INTEGER, Nullable]
    - tot_pymnt [INTEGER, Nullable]
    - tot_pymnt_am [NUMBER (10, 2), Nullable]
    - pay_by_phone [CHAR (2), Nullable]
    - elec_pymnt [CHAR (2), Nullable]
    - pay_in_bank [CHAR (2), Nullable]
    - pay_by_check [CHAR (2), Nullable]
    - pay_by_othr [CHAR (2), Nullable]
    - last_12m_trans_ct [INTEGER, Nullable]
    - bustout [CHAR (2), Nullable]

Y-var

But which of the **X**-Variables should I use when creating my model? Use Correlation to select the best Variables for the **Y**-variable

Lab 1b: Examine the Data with VAL

```
call TRNG_XSP.td_analyze (
       'values',
       'database = TRNG_TDU_TD01;
       tablename = bustout_train;
       columns = all'
       );
```

- tot_pymnt_am [NUMBER (10, 2), Nullable]
- pay_by_phone [CHAR (2), Nullable]
- elec_pymnt [CHAR (2), Nullable]
- pay_in_bank [CHAR (2), Nullable]
- pay_by_check [CHAR (2), Nullable]
- pay_by_othr [CHAR (2), Nullable]
- last_12m_trans_ct [INTEGER, Nullable]
- bustout [CHAR (2), Nullable]

Note 11 columns are 100% null. Let's create a new table without these useless columns. Saves time. And 'bustout' is not numeric

			Nulls
credit_util_prior_5_mth	NUMBER(12,5)	650000	0
days_since_lstcash	INTEGER	650000	650000
days_since_lst_pymnt	INTEGER	650000	0
elec_pymnt	CHAR(1) CHARACTE...	650000	0
last_12m_trans_ct	INTEGER	650000	0
maxamt_e_mt_v7day	NUMBER(10,2)	650000	0
max_utilization_05_mth	NUMBER(12,5)	650000	0
num_pur_lst_60_days	INTEGER	650000	0
num_pur_lst_7_days	INTEGER	650000	0
num_pymnt_lst_60_d...	INTEGER	650000	0
num_pymnt_lst_7_days	INTEGER	650000	0
pay_by_check	CHAR(1) CHARACTE...	650000	0
pay_by_othr	CHAR(1) CHARACTE...	650000	0
pay_by_phone	CHAR(1) CHARACTE...	650000	0
pay_in_bank	CHAR(1) CHARACTE...	650000	0
pct_line_paid_lst_30_...	NUMBER(12,5)	650000	650000
pct_line_paid_lst_7_d...	NUMBER(12,5)	650000	650000
pct_line_pur_lst_30_d...	NUMBER(12,5)	650000	650000
pct_line_pur_lst_7_days	NUMBER(12,5)	650000	650000
set_id	INTEGER	650000	0
times_nsf	INTEGER	650000	0
totcash_to_line_v7day	NUMBER(12,5)	650000	650000
totpmt_to_line_v7day	NUMBER(12,5)	650000	650000
totpurcash_to_line_v7...	NUMBER(12,5)	650000	650000
totpur_to_line_v7day	NUMBER(12,5)	650000	650000
tot_pymnt	INTEGER	650000	650000
tot_pymnt_am	NUMBER(10,2)	650000	650000
tot_pymnt_chnl	INTEGER	650000	0

Lab 1c: Data Cleaning


```
CREATE TABLE bust_out_int
AS (SELECT acct_no, as_of_dt_day, avg_pmt_05_mth, max_utilization_05_mth,
maxamt_epmt_v7day, times_nsf, credit_util_cur_mth, credit_util_prior_5_mth,
credit_util_cur_to_prior_ratio, days_since_lst_pymnt, num_pymnt_lst_7_days,
num_pymnt_lst_60_days, num_pur_lst_7_days, num_pur_lst_60_days, tot_pymnt_chnl,
pay_by_phone, elec_pymnt, pay_in_bank, pay_by_check, pay_by_othr, last_12m_trans_ct,
bustout
FROM TRNG_TDU_TD01.bustout_train) WITH DATA;

ALTER table bust_out_int ADD bustout1 int;

UPDATE bust_out_int SET bustout1=1 WHERE bustout = 'Y'; --147212 fraud transactions;

UPDATE bust_out_int SET bustout1=0 WHERE bustout = 'N'; --502788 good transactions;
```

Lab 2a: Best X-variables to Predict 'Fraud' Y-variable


```
call TRNG_XSP.td_analyze (
      'matrix',
      'database=TRNG_TDU_TD01;
      tablename = bust_out_int;
      nullhandling = zero;
      columnstoexclude = bustout;
      outputdatabase=TRNG_TDU_TD01;
      columns = allnumeric;
      outputtablename = matrix_1;
      overwrite=true;
      matrixtype = COR;'
      );

SELECT * FROM matrix_1 WHERE rowname = 'bustout1';
```

Lab 2b: Best X-variables to Predict 'Fraud' Y-variable

Output

avg_pmt_05_mth	max_utilization_05_mth	maxamt_epmt_v7day	times_nsf	credit_util_cur_mth
0.4438601529391545	0.4434656635477519	0.0762655749550572 3	0.3180094648484720 3	0.569707240621492

credit_util_prior_5_mth	credit_util_cur_to_prior_ratio	days_since_lst_pymnt	num_pymnt_lst_7_days	num_pymnt_lst_60_days
0.5356473567779294	0.1574168693078487 6	0.631220222889617 3	-0.1911918637108066 5	-0.7369479625378376

num_pur_lst_7_days	num_pur_lst_60_days	tot_pymnt_chnl	last_12m_trans_ct	bustout1
-0.3894387769638673	-0.562884079654556	-0.0620840631744045 6	-0.4491303855229410 4	1

Lab 2c: Find Highly Correlated X Variables


```
call TRNG_XSP.td_analyze (
      'matrix',
      'database=TRNG_TDU_TD01;
      tablename = bust_out_int;
      nullhandling = zero;
      columnstoexclude = bustout;
      outputdatabase=TRNG_TDU_TD01;
      columns = allnumeric;
      outputtablename = matrix_2;
      overwrite=true;
      matrixtype = COR;'
      );

SELECT * FROM matrix_2 ORDER BY 1;
```

Lab 2c: Find Highly Correlated X Variables (cont.)

rowname	avg_pmt_05_mth	max_utilization_05_mth	maxamt_epmt_v7day	times_nsf	credit_util_cur_mth	credit_util_prior_5_mth	credit_util_cur_to_prior_ratio
avg_pmt_05_mth	1	0.6266908588912518	0.12459377564726…	-0.02329…	0.580614279297…	0.6713480952149173	-0.006578877286627937
max_utilization_05_mth	0.6266908588912…	1	0.09267783232024…	0.197332…	0.865190317019…	0.9071613574129398	0.11619198393929225
maxamt_epmt_v7day	0.1245937756472…	0.0926778323202464[illegible]	1	0.032011…	0.091487388592…	0.09111741999056991	0.020047386503694517
times_nsf	-0.023293186639…	0.197332970482[illegible]15	0.03201187594395…	1	0.237625807888…	0.02111713344875…	0.3623326906166023
credit_util_cur_mth	0.5806142792974…	0.8651903170192814	0.09148738859263…	0.237625…	1	0.8896134318984398	0.39673885749208654
credit_util_prior_5_mth	0.6713480952149…	0.9071613574129398	0.09111741999056…	0.021117…	0.889613431898…	1	0.027331628558587655
credit_util_cur_to_prior…	-0.006578877286…	0.116191983939[illegible]225	0.02004738650369…	0.362332…	0.396738857492…	0.027331628558587…	1
days_since_lst_pymnt	0.4746842099630…	0.4812371279659[illegible]7	0.08064867839484…	-0.01022…	0.572244741396…	0.6539890052179734	0.009638791236523371
num_pymnt_lst_7_days	-0.128034152289…	-0.139345762894130…	-0.3551936893748…	-0.07191…	-0.179236327513…	-0.1633747263238578	-0.0643767550041322
num_pymnt_lst_60_days	-0.521454327409…	-0.508430965694299	-0.0982595909135…	-0.16329…	-0.638505089963…	-0.6465242684455239	-0.0848140211671301
num_pur_lst_7_days	-0.559446303109…	-0.475287092386910…	-0.0645670742862…	-0.14103…	-0.469723811921…	-0.4677668370558938	-0.07625904281344013
num_pur_lst_60_days	-0.893973699097[illegible]	-0.6599024237471681	-0.1276880818090…	-0.13262…	-0.634632762630…	-0.6767402249964231	-0.025800367348962336
tot_pymnt_chnl	-0.04290811194[illegible]	-0.045767643482233…	-0.1139800[illegible]357…	-0.02885…	-0.054300989216…	-0.05216948337923…	-0.013785583719157057
last_12m_trans_ct	-0.777140142612…	-0.4254951097482098	-0.09842[illegible]0001…	-0.05651…	-0.424487679292…	-0.46575867045334…	-0.02236742536806076
bustout1	0.4438601529391…	0.4434656635477519	0.07626557495505…	0.318009…	0.569707240621…	0.5356473567779294	0.15741686930784876

rowname	days_since_lst_pymnt	num_pymnt_lst_7_days	num_pymnt_lst_60_days	num_pur_lst_7_days	num_pur_lst_60_days	tot_pymnt_chnl	last_12m_trans_ct	bustout1
avg_pmt_05_mth	0.4746842099630584	-0.12803415228988166	-0.5214543274099868	-0.5594463031094…	-0.8939736990973739	-0.0429081194594…	-0.7771401426123…	0.443860152…
max_utilization_05_mth	0.48123712796597967	-0.13934576289413042	-0.508430965694299	-0.4752870923869…	-0.65[illegible]24237471681	-0.0457676434822…	-0.4254951097482…	0.443465663…
maxamt_epmt_v7day	0.08064867839484845	-0.35519368937481127	-0.09825959091350217	-0.0645670742862…	-0.12768[illegible]08180905528	-0.1139800460535…	-0.0984213240001…	0.076265574…
times_nsf	-0.01022920578679…	-0.07191631128842034	-0.16329974232791927	-0.1410332082845…	-0.13262595935746727	-0.0288528137765…	-0.0565184066400…	0.318009464…
credit_util_cur_mth	0.5722447413966076	-0.1792363275136121	-0.6385050899631075	-0.4697238119210…	-0.6346327626305659	-0.0543009892167…	-0.4244876792928…	0.569707240…
credit_util_prior_5_mth	0.6539890052179734	-0.1633747263238578	-0.6465242684455239	-0.4677668370558…	-0.6767402249964231	-0.0521694833792…	-0.4657586704533…	0.535647356…
credit_util_cur_to_prior…	0.009638791236523…	-0.0643767550041322	-0.0848140211671301	-0.0762590428134…	-0.025800367348962…	-0.0137855837191…	-0.0223674253680…	0.157416869…
days_since_lst_pymnt	1	-0.2269893243241512	-0.7199812832877617	-0.3588860775545…	-0.5122787165878029	-0.0774716362734…	-0.3705872048585…	0.631220222…
num_pymnt_lst_7_days	-0.2269893243241512	1	0.27530392551957134	0.08772695217100…	0.1496081613243494	0.32014173090228…	0.09434386771227…	-0.191191863…
num_pymnt_lst_60_days	-0.7199812832877617	0.27530392551957134	1	0.39708075140969…	0.650615541813501	0.13780113122771…	0.46066407760254…	-0.73694796…
num_pur_lst_7_days	-0.35888607755[illegible]2…	0.08772695217100417	0.39708075140969146	1	0.6522535936911011	-0.0553590127417…	0.458051763555451	-0.38943877…
num_pur_lst_60_days	-0.5122787165878[illegible]29	0.1496081613243494	0.650615541813501	0.6522535936911011	1	0.04738854537712…	0.7460785051572284	-0.56288407…
tot_pymnt_chnl	-0.07747163627340…	0.32014173090228343	0.13780113122771093	-0.0553590127417…	0.0473885453771271	1	0.03247068071186…	-0.06208406…
last_12m_trans_ct	-0.3705872048585899	0.09434386771227411	0.46066407760254013	0.458051763555451	0.7460785051572284	0.03247068071186…	1	-0.44913038…
bustout1	0.6312202228896173	-0.19119186371080665	-0.7369479625378376	-0.3894387769638…	-0.5[illegible]84079654556	-0.0620840631744…	-0.4491303855229…	1

Lab 2de: View the Partitioned Data

Goal: Partition Data by 'YEARID' and Correlate 'hits', 'runs' and 'era' (X-vars) to 'wins' (Y-var)

```
SELECT yearid, w, h, r, era
FROM teams_prior2012
WHERE yearid > 2007
ORDER BY yearid;
```

yearid	teamid	w	h	r	era
2009	PIT	62	1364	636	4.5900
2009	KCA	65	1432	686	4.8300
2009	NYA	103	1604	915	4.2800
2009	CHN	83	1398	707	3.8400
2010	NYN	79	1361	656	3.7300
2010	SFN	92	1411	697	3.3600
2010	BAL	66	1440	613	4.5900
2010	SEA	61	1274	513	3.9500

Lab 2e: Best X-variables by Year Partition

```
call TRNG_XSP.td_analyze (
       'matrix',
       'database=TRNG_TDU_TD01;
       tablename = teams_prior2012;
       where = yearid > 2007;
       outputdatabase=TRNG_TDU_TD01;
       columns = w, h, r, era;
       groupby = yearid;
       outputtablename = matrix_1;
       overwrite=true;
       matrixtype = COR;'
       );
```

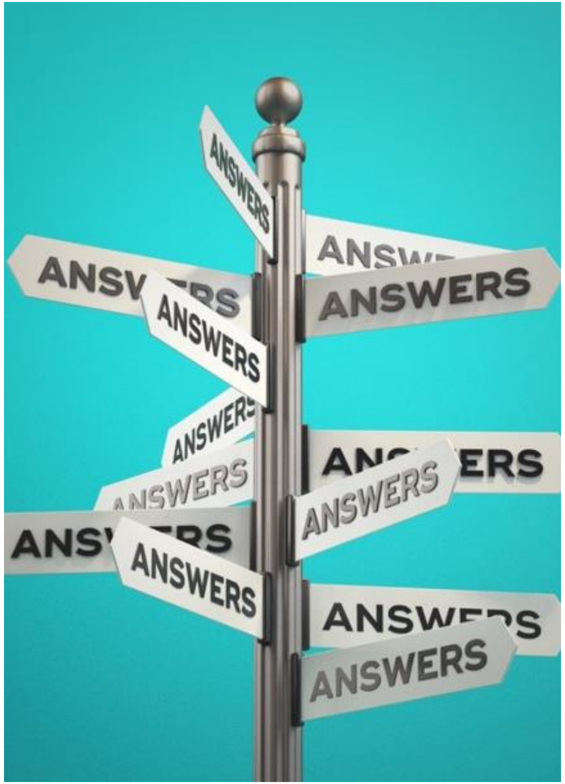
yearid	rownum	rowname	w	h	r	era
2008	2	h	0.246826...	1	0.671850...	0.220477...
2008	1	w	1	0.246826...	0.587016...	-0.668721...
2008	4	era	-0.66872...	0.220477...	0.069691...	1
2008	3	r	0.587016...	0.671850...	1	0.069691...
2009	2	h	0.438749...	1	0.764469...	0.078437...
2009	3	r	0.611273...	0.764469...	1	0.073179...
2009	4	era	-0.63216...	0.0784[illegible]...	0.073179...	1
2009	1	w	1	0.43874[illegible]	0.611273...	-0.632168...
2010	1	w	1	0.446067...	0.784809...	-0.687623...
2010	4	era	-0.68762...	0.083004...	-0.16982...	1
2010	3	r	0.784809...	0.683631...	1	-0.169824...
2010	2	h	0.446067...	1	0.683631...	0.083004...
2011	3	r	0.600808...	0.801210...	1	0.142759...
2011	4	era	-0.61474...	0.323076...	0.142759...	1
2011	2	h	0.297659...	1	0.801210...	0.323076...
2011	1	w	1	0.297659...	0.600808...	-0.614743...

Did the Correlation values change from Year to Year? The answer is 'Yes'.

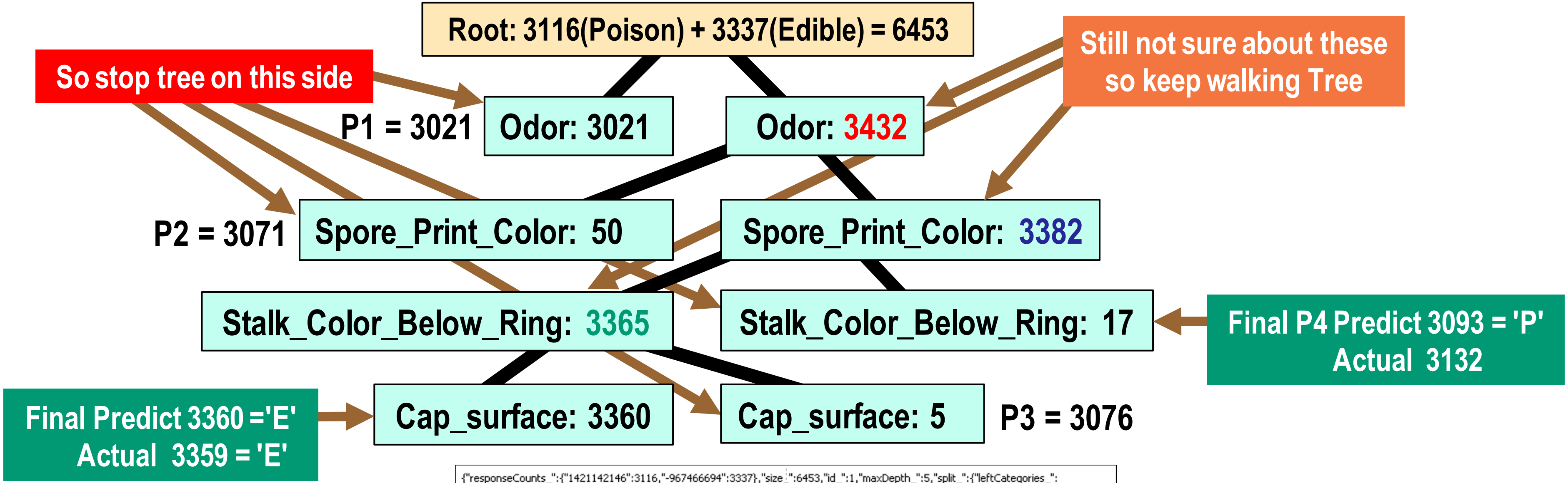
In 2008, ERA (-0.669) was most highly Correlated with Wins.
But in Year 2010, Runs (0.785) was most Correlated

Current Topic – Decision Forest Trees (with ConfusionMatrix)

- Predictive Analytics Overview
- Correlation
- **Ensemble Functions**
  - **Decision Forest Trees (with ConfusionMatrix)**
  - XGBoost
- Summary



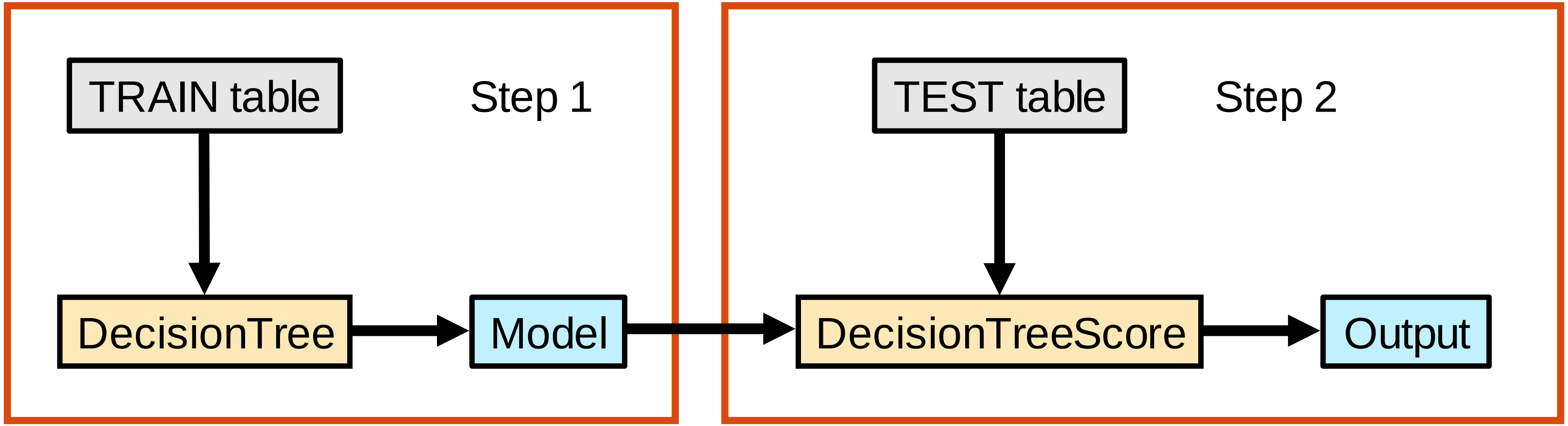
Description – How Decision Tree Works



{"responseCounts_":{"1421142146":3116,"-967466694":3337},"size_":6453,"id_":1,"maxDepth_":5,"split_":{"leftCategories_":
[191545620,55990575,1421142146,-1877468854,1063097864,-342070737],"rightCategories_":[1210518909,1184742640,-
1563381124],"attr_":"odor","score_":0.028628649618886715,"type_":"CLASSIFICATION_CATEGORICAL_SPLIT","leftNodeSize_":3021,"ri
ghtNodeSize_":3432},"leftChild_":{"label_":"p","size_":3021,"id_":2,"maxDepth_":0,"nodeType_":"CLASSIFICATION_LEAF"},"rightChild_":
{"responseCounts_":{"1421142146":95,"-967466694":3337},"size_":3432,"id_":3,"maxDepth_":4,"split_":{"leftCategories_":
[1906364349],"rightCategories_":[191545620,1210518909,-1853091852,-994553553,-1567496736,-1111134317,-
707696126,1489429165],"attr_":"spore_print_color","score_":0.02587484957384421,"type_":"CLASSIFICATION_CATEGORICAL_SPLIT","
leftNodeSize_":50,"rightNodeSize_":3382},"leftChild_":"label_":"p","size_":50,"id_":6,"maxDepth_":0,"nodeType_":"CLASSIFICATION_LEA
F"},"rightChild_":{"responseCounts_":{"1421142146":45,"-967466694":3337},"size_":3382,"id_":7,"maxDepth_":3,"split_":
{"leftCategories_":[1210518909,-371515505,1421142146,-994553553,-967466694,-707696126],"rightCategories_":
[191545620],"attr_":"stalk_color_below_ring","score_":0.01642046917383611,"type_":"CLASSIFICATION_CATEGORICAL_SPLIT","leftNod
eSize_":3365,"rightNodeSize_":17},"leftChild_":{"responseCounts_":{"1421142146":28,"-
967466694":3337},"size_":3365,"id_":14,"maxDepth_":2,"split_":{"leftCategories_":[191545620,55990575,-
342070737],"rightCategories_":371515505],"attr_":"cap_surface","score_":0.013576558409396424,"type_":"CLASSIFICATION_CATEGOR
ICAL_SPLIT","leftNodeSize_":3360,"rightNodeSize_":5},"leftChild_":"label_":"e","size_":3360,
SIFICATION_LEAF"},"rightChild_":"label_":"p","size_":5,"id_":29,"maxDepth_":0,"nodeType_
LASSIFICATION_NODE"},"rightChild_":"label_":"p","size_":17,"id_":15,"maxDepth_":0,"nodeT
_":"CLASSIFICATION_NODE"},"nodeType_":"CLASSIFICATION_NODE"},"nodeType_":"CLASS

Stops at 'Cap Surface' because could no longer find separation between 'P' and 'E'

DecisionTree (VAL) – Workflow



DecisionTree – Overview

Currently, the Teradata Warehouse Miner External Stored Procedure provides decision trees for classification models. They are built largely on the techniques described in [Quinlan] and as such, splits using information gain ratio are provided. Pruning is also provided, also using the gain ratio technique. The concept of Information gain ratio is simple - the more you know about a topic, the less new information you are apt to get about it. To be more concise: If you know an event is very probable, it is no surprise when it happens - that is, it gives you little information that it actually happened. Taking this a bit further, we can formulate that the amount of information gained is inversely proportional to the probability of an event happening. Given that entropy refers to the probability of an event occurring, we can also say that as the entropy increases, the information gain decreases. A decision tree scoring function is provided to score and/or evaluate a decision tree model.

DecisionTree – Syntax


```
call ${XSPDB}.td_analyze('decisiontree',
                         'database=database;
                          tablename=table;
                          columns=column1,column2,…;
                          dependent=response;
                          min_records=2;
                          max_depth=5;
                          binning=false;
                          algorithm=gainratio;
                          pruning=gainratio;
                          outputdatabase=database;
                          outputtablename=DecisionTree1;
                          operatordatabase=VAL_database;');
```

DecisionTree – Required Arguments

- **Columns:** (**X**-var). Specify the names of columns that contain numeric predictor variables (which must be numeric values) and specify the names of the columns that contain the categorical predictor variables (which can be either numeric or VARCHAR values).
- **Database:** Specify the database that has the input file
- **DecisionTree:** Identifies the type of function being performed
- **Dependent:** Specify the name of the column that contains the response variable (that is, what you want to predict). (**Y**-var)
- **TableName:** Specify the table to build the model from

DecisionTree – Optional Arguments

- **Algorithm:** The algorithm the decision tree uses during building. Currently this option only allows gainratio.
- **Binning:** Option to automatically Bincode the continuous independent variables. Continuous data is separated into one hundred bins when this option is selected. If the variable has fewer than one hundred distinct values, this option is ignored. Default is false.
- **Max_Depth:** Maximum number of levels the tree can grow. The default is 100.
- **Min_Records:** Specifies how far the decision tree can split. Unless a node is pure (meaning it has only observations with the same dependent value) it splits if each branch that can come off this node contains at least this many observations. The default is a minimum of two cases for each branch.

DecisionTree – Optional Arguments (cont.)

- **OperatorDatabase:** The database where the table operators called by td_analyze reside.
- **OutputDatabase:** The database containing the resulting output table when outputstyle=table or view.
- **OutputTableName:** The name of the output table representing the decision tree model.
- **Overwrite:** When overwrite is set to true (default), the output tables are dropped before creating new ones.
- **Pruning:** Determines the style of pruning to use after the tree is fully built. The default option is gainratio. The only other option at this time is none which does no pruning of the tree.

DecisionTree – Optional Arguments (cont.)

- **ColumnsToExclude:** If a column specifier such as all is used in the columns' parameter, the columnstoexclude parameter may be used to exclude specific columns from the analysis. For convenience, when the columnstoexclude parameter is used, dependent variable and group by columns, if any, are automatically excluded as input columns and do not need to be included as columnstoexclude.

Lab 3a: View the Data

- Here's the data we'll be using. We will be predicting Y-variable 'homestyle'. Since Y-variable is VARCHAR, TreeType will default to Classification
- There are 12 X-variables (6 numerical predictors and 6 categorical predictors)
  - Numerical: price, lotsize, bedrooms, bathrms, stories, garagepl
  - Categorical: driveway, recroom, fullbase, gashw, airco, prefarea

```
SELECT *
FROM
housing_train_updated;
```

	X	X	X	X	X	X	X	X	X	X	X	X	Y-var
sn	price	lotsize	bedrooms	bathrms	stories	driveway	recroom	fullbase	gashw	airco	garagepl	prefarea	homestyle
86	57000	6480	3	1	2	no	no	no	no	yes	1	no	Eclectic
87	60000	5850	2	1	1	yes	yes	yes	no	no	2	no	Eclectic
88	78000	3150	3	2	1	yes	yes	yes	no	yes	0	no	Eclectic
89	35000	3000	2	1	1	yes	no	no	no	no	1	no	Classic
90	44000	3090	2	1	1	yes	yes	yes	no	no	0	no	Classic
91	47000	6060	3	1	1	yes	yes	yes	no	no	0	no	Classic
92	58000	5900	4	2	2	no	no	yes	no	no	1	no	Eclectic
93	163000	7420	4	1	2	yes	yes	yes	no	yes	2	no	bungalow
94	128000	8500	3	2	4	yes	no	no	no	yes	2	no	bungalow
95	123500	8050	3	1	1	yes	yes	yes	no	yes	1	no	bungalow
96	39000	6800	2	1	1	yes	no	no	no	no	0	no	Classic
97	53900	8250	3	1	1	yes	no	no	no	no	2	no	Eclectic
98	59900	8250	3	1	1	yes	no	yes	no	no	3	no	Eclectic

Lab 3b: DecisionTree – Create 'Classification' Model

teradata.


```
call TRNG_XSP.td_analyze('decisiontree',
                         'database=TRNG_TDU_TD01;
                          tablename=housing_train_updated;
columns=price,lotsize,bedrooms,bathrms,stories,garagepl,driveway,
                     recroom,fullbase,gashw,airco,prefarea;
                          dependent=homestyle;
                          min_records=3;
                          max_depth=12;
                          binning=false;
                          algorithm=gainratio;
                          pruning=gainratio;
                          outputdatabase=TRNG_TDU_TD01;
                          outputtablename=DecisionTree1;
                           overwrite=true;
                          operatordatabase=TRNG_XSP;');

SELECT * FROM DecisionTree1;
```


```
  <Value value="Eclectic"/>
 </DataField>
</DataDictionary>
<TreeModel modelName="Tree Model" algorithmName="Gain Ratio" >
 <Extension name="nodesBeforePruning" value="23"/>
 <Extension name="nodesAfterPruning" value="19"/>
 <MiningSchema>
  <MiningField name="price"/>
  <MiningField name="lotsize"/>
  <MiningField name="bedrooms"/>
  <MiningField name="bathrms"/>
  <MiningField name="stories"/>
  <MiningField name="garagepl"/>
  <MiningField name="driveway"/>
  <MiningField name="recroom"/>
  <MiningField name="fullbase"/>
  <MiningField name="gashw"/>
  <MiningField name="airco"/>
  <MiningField name="prefarea"/>
  <MiningField name="homestyle" usageType="predicted"/>
 </MiningSchema>
 <Node score="Eclectic" recordCount="738">
  <True/>
  <ScoreDistribution value="bungalow" recordCount="86"/>
  <ScoreDistribution value="Classic" recordCount="211"/>
  <ScoreDistribution value="Eclectic" recordCount="441"/>
  <Node score="Classic" recordCount="140">
```

DecisionTreeScore – Syntax

DecisionTreeScore: In order to deploy the gain ratio decision tree model created above, a companion decision tree scoring function is provided to score and/or evaluate a decision tree model.

```
call TRNG_XSP.td_analyze('decisiontreescore',
                         'database=database;
                          tablename=table;
                          modeldatabase=database;
                          modeltablename=DecisionTree1;
                          outputdatabase=database;
                          predicted=Predicted;
                          retain=column1,column2,..;
                          outputtablename=DecisionTreeScore1;
                          scoringmethod=scoreandevaluate;
                          includeconfidence=true;');
```

DecisionTreeScore – Arguments

- **Database:** Specify the database that has the input file.
- **ModelDatabase:** The database containing the table representing the decision tree model input to the analysis.
- **ModelTableName:** The table containing the decision tree model in PMML format that is used to score the data. It must reside in the database indicated by the modeldatabase parameter.
- **OutputDatabase:** The database containing the output table.
- **OutputTableName:** The output table containing the predicted values of the dependent variable. It must reside in the database indicated by the outputdatabase parameter.
- **TableName:** The table containing the columns to analyze, representing the dependent and independent variables in the analysis. It must reside in the database indicated by the database parameter.

DecisionTreeScore – Optional Arguments

- **Confusionmatrix:** A table delivered in the function's XML output string, displaying counts of predicted versus actual values of the dependent variable of the decision tree model.
- **Gensqlonly:** When true, the SQL for the requested function is returned as a result set but not run. When not specified or set to false, the SQL is run but not returned.
- **Includeconfidence:** If selected, the output table will contain a column indicating how likely it is, for a particular leaf node on the tree, that the prediction is correct.
- **Index:** By default, the primary index columns of the score output table are the primary index columns of the input table. This parameter allows the user to specify one or more columns for the primary index of the score output table.

DecisionTreeScore – Optional Arguments (cont.)

- **Overwrite:** When overwrite is set to true or not set, the output table is dropped before creating a new one.
- **Predicted:** If the ‘scoringmethod’ parameter is set to ‘score’ or ‘scoreandevaluate’, the name of the predicted value column is entered here. If not entered here, the name of the dependent column in the input table is used.
- **Retain:** One or more columns from the input table can optionally be specified here to be passed along to the score output table.
- **Samplescoresize:** When a scoring function produces a score table, the user has the option to view a sample of the rows using the "samplescoresize=n" parameter, where n is an integer number of rows to view in a result set. Cases where a sample is not returned include when you only generating SQL and when you are only evaluating (i.e., not scoring). By default, a sample of output score rows is not returned.

DecisionTreeScore – Optional Arguments (cont.)

- **Scoringmethod:** Three scoring methods are available as outlined below. By default, the model is scored but not evaluated, as requested in this manner: scoringmethod=score.
  - score
  - evaluate
  - scoreandevaluate
- **Targetedvalue:** If selected, the output table will contain a column indicating how likely it is, for a particular leaf node and targeted value of a predicted result with only two values, that the prediction is correct.

Lab 4a: Using VAL 'DecisionTreeScore'


```
call TRNG_XSP.td_analyze('decisiontreescore',
                         'database=TRNG_TDU_TD01;
                          tablename=housing_test_updated;
                          modeldatabase=TRNG_TDU_TD01;
                          modeltablename=DecisionTree1;
                          outputdatabase=TRNG_TDU_TD01;
                          predicted=predicted_homestyle;
                          retain=homestyle;
                          outputtablename=DecisionTreeScore1;
                         overwrite=true;
                          includeconfidence=true;
                          scoringmethod=scoreandevaluate;
                          includeconfidence=true;');
```

Lab 4b: View Prediction Results and Accuracy

```
SELECT * FROM TRNG_TDU_TD01.DecisionTreeScore1;
```

Actual | Predict

sn	homestyle	predicted_homestyle	_tm_confidence
13	Classic	Classic	1
16	Classic	Classic	1
25	Classic	Classic	1
38	Eclectic	Eclectic	0.98666666666...
53	Eclectic	Eclectic	0.98666666666...
104	bungalow	Classic	1
111	Classic	Classic	1
117	Eclectic	Eclectic	0.98666666666...
132	Classic	Classic	1
140	Classic	Classic	1
142	Classic	Classic	1
157	Eclectic	Eclectic	0.98666666666...
161	Eclectic	Eclectic	0.98666666666...
162	bungalow	bungalow	0.97826086956...
176	Eclectic	Eclectic	0.98666666666...

95% accurate (Results may vary)

```
SELECT
(SELECT cast(count(*) as dec(4,2))
 FROM DecisionTreeScore1
WHERE homestyle =
predicted_homestyle)/
(SELECT cast(count(*) as dec(4,2))
 FROM DecisionTreeScore1);
```

(Count(*)/Count(*))
.95

Lab 4b: View Prediction Results and Accuracy (cont.)

teradata.


```
call TRNG_XSP.td_analyze('histogram',

'database=TRNG_TDU_TD01;

tablename=DecisionTreeScore1;

columns=homestyle,predicted_homestyle;
                          quantiles=10;
                          style=crosstab');
```

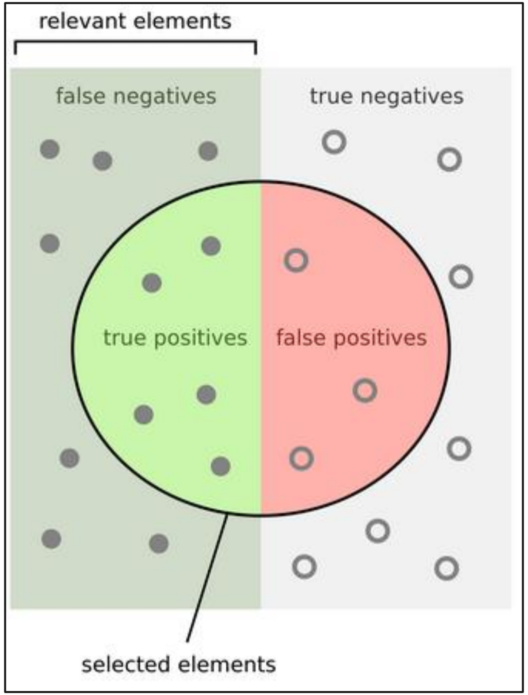
xbin_homestyle	xbeg_homestyle	xend_homestyle	xbin_predicted_homestyle	xbeg_predicted_homestyle	xend_predicted_homestyle	xcnt	xpct
1	bungalow	bungalow	1	bungalow	bungalow	13	16.049382716049383
1	bungalow	bungalow	2	Classic	Classic	2	2.4691358024691357
2	Classic	Classic	1	bungalow	bungalow	1	1.2345679012345678
2	Classic	Classic	2	Classic	Classic	25	30.864197530864196
2	Classic	Classic	6	Eclectic	Eclectic	1	1.2345679012345678
6	Eclectic	Eclectic	6	Eclectic	Eclectic	39	48.148148148148145

Description – ConfusionMatrix Function

- Show how often a classification algorithm correctly classifies items
- Three tables are Output:
  - A confusion matrix which shows performance of the algorithm
  - A table of overall statistics
  - A table of statistics for each class

	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

TN = True Negative
FP = False Positive
FN = False Negative
TP = True Positive



ConfusionMatrix in Decision Tree Scoring Summary

The output table has the actual home style in 'homestyle' and the prediction in predicted_homestyle

Better looking report than histogram crosstab

```
call TRNG_XSP.td_analyze ('report’,
             'database=TRNG_TDU_TD01;
             tablename=DecisionTreeScore1;
             analysistype=decisiontreescore');
```

Decision Tree Scoring Summary

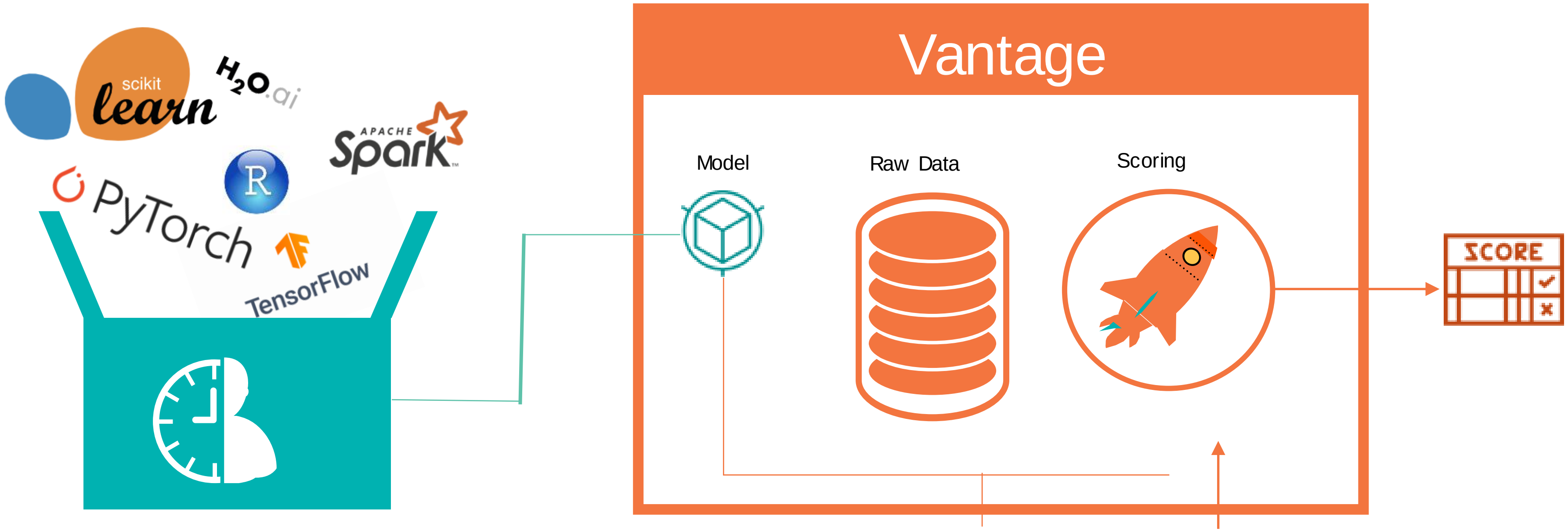
Confusion Matrix

-	Actual bungalow	Actual Classic	Actual Eclectic	Correct	Incorrect
Predicted bungalow	13.00/16.05%	1.00/1.23%	0.00/0.00%	13.00/16.05%	1.00/1.23%
Predicted Classic	2.00/2.47%	25.00/30.86%	0.00/0.00%	25.00/30.86%	2.00/2.47%
Predicted Eclectic	0.00/0.00%	1.00/1.23%	39.00/48.15%	39.00/48.15%	1.00/1.23%

Bring Your Own Model Vision

Customers can build models in any language using most-popular tools or platforms and score them at scale in Vantage with minimal data movement.

- Import existing Machine Learning models to Vantage



Bring Your Own Model (BYOM) Workflow

Allow Data Scientists freedom to use preferred tools

1. Prepare/Transform Data in Vantage

Vantage System

2. Transformed Training Dataset Exported for Model Training

3. Train/Build model using any preferred tools

4. Import the model

5. Score/Predict entire data set at scale in Vantage

Data Science Workflow for Vantage BYOM Feature

MODEL OPERATIONALIZATION

Data Preparation

- Explore & profile data
- Clean, normalize & transform data
- Derive new variables
- Create Analytic Dataset or Feature Store

Model Creation

- Export sample data to modelling system
- Identify Machine Learning algorithm
- Fit Model
- Create predictive model using Analytic dataset for training

Model Scoring

- Export model to standard model format (PMML, MLeap, ONNX, etc.)
- Import model to Vantage
- Score model using BYOM Predict Function
- Evaluate Quality of Predictions

Environment Re-creation

- Transform production dataset
- Create Analytic Dataset or Feature Store
- Upload Model to production system

Model Scoring

- Schedule batch and/or on-demand Scoring job
- Setup downstream Business reports and program actions derived from Predicted values
- Periodically review and refine model as needed

Lab 5a: Jupyter Load Libraries and Java

Install these packages from the Launcher Page. Choose File menu, New Launcher
From Launcher choose Terminal, and enter these commands one at a time:

```
pip install sklearn2pmml
pip install setuptools-git
pip install cmdstanpy==0.4
pip install install-jdk
pip install xgboost
# Run code to the left to find Java location
cd /opt/conda/bin
ln -s /home/jovyan/.jdk/jdk-11.0.12+7/bin/java  java
```

XGBoost not needed for Decision Tree, used in following lab

To find Java location:

```
python3
>>>import jdk
>>> jdk.install('11')
# Wait for location of jdk to display
/home/jovyan/.jdk/jdk-11.0.12+7
>>>exit()
```

After done with above commands restart kernel

Lab 5b: View the Data

- Here's the data we'll be using. We will be predicting Y-variable 'homestyle'. This is a classification model
- There are 12 X-variables, all numeric
- price, lotsize, bedrooms, bathrms, stories, garagepl
- 0=no, 1=yes: driveway, recroom, fullbase, gashw, airco, prefarea

```
train_df = DataFrame.from_query("select *
FROM TRNG_TDU_TD01.housing_train_int")
traid_pd = train_df.to_pandas()
traid_pd
```

Home Style
0 = bungalow
1 = classic
2 = eclectic

	X	X	X	X	X	X	X	X	X	X	X	X	Y-var	Not used
	price	lotsize	bedrooms	bathrms	stories	driveway	recroom	fullbase	gashw	airco	garagepl	prefarea	homestyle	sell_date
sn														
1203	175000.0	3480.0	3	1	1	0	0	0	0	1	0	0	1	2019-01-01
265	50000.0	3640.0	2	1	1	1	0	0	0	0	1	0	1	1984-01-01
61	48000.0	4120.0	2	1	2	1	0	0	0	0	0	0	1	1984-01-01
122	80000.0	10500.0	4	2	2	1	0	0	0	0	1	0	2	198[illegible]01
326	99000.0	8880.0	3	2	2	1	0	1	0	1	1	0	2	1984-[illegible]-01

Lab 6a: DecisionTree – Create 'Classification' Model

teradata.

Define X and Y variables

```
X = traid_pd[['price','lotsize','bedrooms', 'bathrms', 'stories', 'garagepl',
'driveway', 'recroom', 'fullbase', 'gashw', 'airco', 'prefarea']]

y = traid_pd[['homestyle']]
```

Pipeline and Export to PMML

```
pipeline = PMMLPipeline([
("classifier", tree.DecisionTreeClassifier())
])
pipeline.fit(X, y.values.ravel())
sklearn2pmml(pipeline, "housing_db_dt_model.pmml", with_repr = True)
```

Lab 6b: DecisionTree – Upload 'Classification' Model

teradata.

Create & Clear model table

```
con.execute("CREATE SET TABLE pmml_models
     (model_id VARCHAR(40) CHARACTER SET LATIN NOT CASESPECIFIC,
      model BLOB(2097088000)) PRIMARY INDEX ( model_id );")
con.execute("delete from pmml_models where model_id = 'housing_db_dt_model'")
```

Uploading the PMML model

```
model_bytes = open("housing_db_dt_model.pmml", "rb").read()
con.execute("insert into mldb.pmml_models  (model_id, model) values(?,?)",
'housing_db_dt_model', model_bytes)
model_list = pd.read_sql("select * from
mldb.pmml_models", con)
model_list
```

	model_id	model
0	iris_db_rf_model	b'<?xml version="1.0" encoding="UTF-8" standal...
1	iris_db_glm_model	b'<?xml version="1.0" encoding="UTF-8" standal...
2	housing_db_dt_model	b'<?xml version="1.0" encoding="UTF-8" standal...
3	iris_db_xgb_model	b'<?xml version="1.0" encoding="UTF-8" standal...
4	iris_rf_class_model	b'<?xml version="1.0" encoding="UTF-8"?>\n<PMM...
5	iris_db_dt_model	b'<?xml version="1.0" encoding="UTF-8" standal...
6	iris_db_naive_bayes_model	b'<?xml version="1.0" encoding="UTF-8" standal...

Lab 6c: DecisionTree – Score 'Classification' Model

teradata.

Clear scoring table

```
con.execute("DROP TABLE housing_df_out;")
```

Call scoring function—PMMLPredict

```
con.execute("CREATE TABLE housing_df_out AS (
SELECT * FROM TRNG_BYOM.PMMLPredict(
    ON TRNG_TDU_TD01.housing_test_int
    ON (select * from pmml_models where model_id='housing_db_dt_model')
DIMENSION
    USING
        Accumulate('sn','homestyle')
) AS dt
) WITH DATA;")
```

sn	homestyle	prediction	json_report
469	2		{"probability(2)":1.0,"probability(1)":0.0,"probability(0)":0.0}
530	0		{"probability(2)":0.0,"probability(1)":0.0,"probability(0)":1.0}
162	0		{"probability(2)":0.0,"probability(1)":0.0,"probability(0)":1.0}
1140	1		{"probability(2)":0.0,"probability(1)":1.0,"probability(0)":0.0}
1527	0		{"probability(2)":0.0,"probability(1)":0.0,"probability(0)":1.0}

Nothing in prediction column?

Lab 6d: DecisionTree – Score 'Classification' Model with Model Output Fields

teradata.

Clear scoring table

```
con.execute("DROP TABLE housing_df_out;")
```

Call scoring function again—PMMLPredict

```
con.execute("CREATE TABLE housing_df_out AS (
SELECT * FROM TRNG_BYOM.PMMLPredict(
    ON TRNG_TDU_TD01.housing_test_int
    ON (select * from pmml_models where model_id='housing_db_dt_model')
DIMENSION
    USING
        Accumulate('sn','homestyle') \
        ModelOutputFields ('probability(2)’,
         'probability(1)', 'probability(0)')
) AS dt
) WITH DATA;")
```

sn	homestyle	prediction	probability(2)	probability(1)	probability(0)
469	2		1.0	0.0	0.0
530	0		0.0	0.0	1.0
162	0		0.0	0.0	1.0
1140	1		0.0	1.0	0.0
1527	0		0.0	0.0	1.0

Lab 6e: DecisionTree – Score 'Classification' Model with Model Output Fields and Fix

Update prediction column

```
con.execute…
('UPDATE housing_df_out SET prediction=2 WHERE "probability(2)" GT 0;')
('UPDATE housing_df_out SET prediction=1 WHERE "probability(1)" GT 0;')
('UPDATE housing_df_out SET prediction=0 WHERE "probability(0)" GT 0;')
```

sn	homestyle	prediction	probability(2)	probability(1)	probability(0)
469	2	2	1.0	0.0	0.0
530	0	0	0.0	0.0	1.0
162	0	0	0.0	0.0	1.0
1140	1	1	0.0	1.0	0.0
1527	0	0	0.0	0.0	1.0

Lab 6e: DecisionTree – Score 'Classification' Model with Model Output Fields and Fix (cont.)

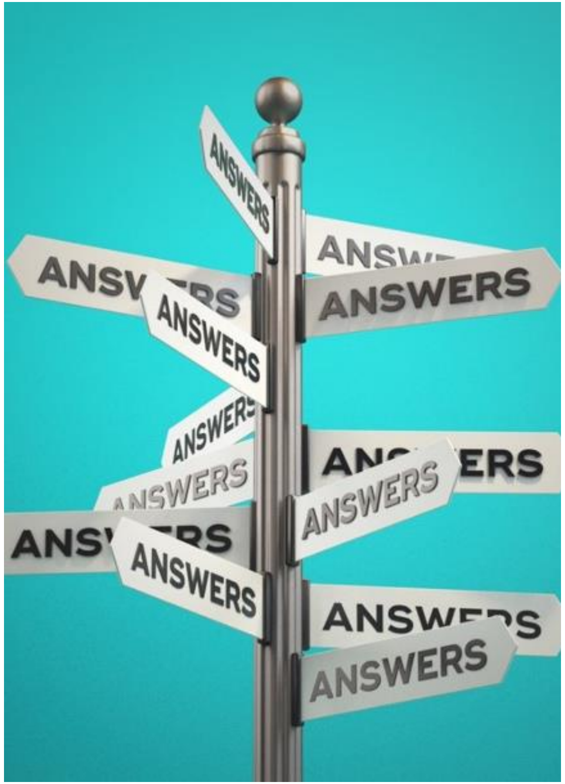
Check Accuracy

```
housing_ac = DataFrame.from_query("SELECT (SELECT count(sn)*1.00
   FROM housing_df_out
   WHERE homestyle = prediction) / (SELECT count(sn)
   FROM housing_df_out) AS PA;")
housing_accr = housing_ac.to_pandas()
housing_accr
```

	PA
0	0.96

Current Topic – XGBoost

- Predictive Analytics Overview
- Correlation
- **Ensemble Functions**
  - Decision Forest Trees (with ConfusionMatrix)
  - **XGBoost**
- Summary



Description – XGBoost

- In Gradient boosting, each iteration fits a Model to the residuals (errors) of the previous iteration. It also provides a general framework for adding a loss function and a regularization term
- **XGBoost** supports both Dense and Sparse data sets

How it Works – Gradient Boosting Process

- The **XGBoost** functions use gradient boosting, which provides a general framework for adding any loss function and applies some optimizations for better scalability
- The statistical framework cast boosting as a numerical optimization problem where the objective is to minimize the loss of the model by adding weak learners using a gradient descent like procedure
- Gradient boosting involves three elements:
  1. A Loss function to be optimized (Measure of how good are model's Coefficients at fitting the underlying data. Coefficients are Measure of variance of Y-variable using the X-variables.)
  2. A weak learner to make predictions
  3. An additive model to add weak learners to minimize the loss function

A **Loss function** is a measure of how good a prediction Model does of being able to predict the expected outcome
https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

Lab 7a: Jupyter Load Libraries and Java

Install these packages from the Launcher Page. Choose File menu, New Launcher
From Launcher choose Terminal, and enter these commands one at a time:

```
pip install sklearn2pmml
pip install setuptools-git
pip install cmdstanpy==0.4
pip install install-jdk
pip install xgboost
# Run code to the left to find Java location
cd /opt/conda/bin
ln -s /home/jovyan/.jdk/jdk-11.0.12+7/bin/java  java
```

To find Java location:

```
python3
>>>import jdk
>>> jdk.install('11')
# Wait for location of jdk to display
/home/jovyan/.jdk/jdk-11.0.12+7
>>>exit()
```

After done with above commands restart kernel

Lab 7d: View the Data

- Here's the data we'll be using. We will be predicting Y-variable bustout1'. This is a classification model. X variables are most correlated.

```
train_df = DataFrame.from_query("select * FROM TRNG_TDU_TD01.bust_out_int")
traid_pd = train_df.to_pandas()
traid_pd
```

- There are three X-variables, all numeric
  - days_since_lst_pymnt
  - num_pymnt_lst_60_days
  - num_pur_lst_60_days

Bustout1
0 = no fraud
1 = fraud

X variables

Y-var

	days_since_lst_pymnt	num_pymnt_lst_60_days	num_pur_lst_60_days	bustout1
acct_no				
257	6	1	6	0
300	8	1	8	0
257	16	1	8	0
300	9	1	8	0
257	23	1	11	0

Lab 8a: XGBoost – Create 'Classification' Model

teradata.

Define X and Y variables

```
X = traid_pd[['days_since_lst_pymnt','num_pymnt_lst_60_days',
'num_pur_lst_60_days' ]]
y=traid_pd[['bustout1']]
```

Pipeline and Export to PMML

```
pipeline = PMMLPipeline([
("classifier", XGBClassifier())
])
pipeline.fit(X, y.values.ravel())
sklearn2pmml(pipeline, "bustout_xgb_model.pmml", with_repr = True)
```

Lab 8b: XGBoost – Upload 'Classification' Model

Clear model table

```
con.execute("delete from pmml_models where model_id = 'bustout_xgb_model'")
```

Uploading the PMML model

```
model_bytes = open("bustout_xgb_model.pmml", "rb").read()
con.execute("insert into pmml_models  (model_id, model) values(?,?)",
'bustout_xgb_model', model_bytes)

model_list = pd.read_sql("select *
      from pmml_models", con)
model_list
```

	model_id	
0	iris_db_rf_model	b'<?xml version="1.0" encoding="UTF-8" standalone="yes"?>\n<PMML
1	bustout_xgb_model	b'<?xml version="1.0" encoding="UTF-8" standalone="yes"?>\n<PMML
2	housing_db_dt_model	b'<?xml version="1.0" encoding="UTF-8" standalone="yes"?>\n<PMML
3	iris_db_glm_model	b'<?xml version="1.0" encoding="UTF-8" standalone="yes"?>\n<PMML
4	iris_rf_class_model	b'<?xml version="1.0" encoding="UTF-8"?>\n<PMML xmlns="
5	iris_db_xgb_model	b'<?xml version="1.0" encoding="UTF-8" standalone="yes"?>\n<PMML
6	iris_db_dt_model	b'<?xml version="1.0" encoding="UTF-8" standalone="yes"?>\n<PMML
7	iris_db_naive_bayes_model	b'<?xml version="1.0" encoding="UTF-8" standalone="yes"?>\n<PMML

Lab 8c: XGBoost – Score 'Classification' Model

Drop scoring table

```
con.execute("DROP TABLE bustout_xgb_out;")
```

Call scoring function—PMMLPredict

```
con.execute("CREATE TABLE bustout_xgb_out AS (
SELECT * FROM TRNG_BYOM.PMMLPredict(
    ON TRNG_TDU_TD01.bustout_test
    ON (select * from pmml_models where model_id='bustout_xgb_model')
DIMENSION
    USING
        Accumulate('acct_no')
        ModelOutputFields ('probability(0)',
       'probability(1)')
) AS dt \
) WITH DATA;")
```

Nothing in prediction column?

	prediction	probability(0)	probability(1)
acct_no			
257		0.892505	0.107495
300		0.970707	0.029293
257		0.857499	0.142501
300		0.998761	0.001239
257		0.887178	0.112822

Lab 8cd: XGBoost – Score 'Classification' Model with Model Output Fields and Fix

Update prediction column

```
con.execute…

('UPDATE bustout_xgb_out SET prediction=0 WHERE "probability(0)" GT
"probability(1)";')

('UPDATE bustout_xgb_out SET prediction=1 WHERE "probability(1)" GT
"probability(0)";')
```

acct_no	prediction	probability(0)	probability(1)
257	0	0.892505	0.107495
300	0	0.970707	0.029293
257	0	0.857499	0.142501
300	0	0.998761	0.001239
257	0	0.887178	0.112822

Lab 8e: XGBoost – Accuracy Table

Create Table with actual bustout for test set

```
con.execute("CREATE MULTISET TABLE bustout_xgb_accuracy AS
(SELECT t.acct_no, t.bustout, p.prediction FROM bustout_xgb_out p,
TRNG_TDU_TD01.bustout_test t
 WHERE t.acct_no = p.acct_no
) WITH DATA;")
```

View the table

```
bustout_ac = DataFrame.from_query("select * FROM
TRNG_TDU_TD01.bustout_xgb_accuracy")
bustout_pda = bustout_ac.to_pandas()
bustout_pda
```

acct_no	bustout	prediction
257	N	0
300	N	0
257	N	0
300	N	0
257	N	0

Bustout and prediction are not the same format

Lab 8f: XGBoost – Accuracy Table Fix

Fix the table

```
con.execute("ALTER table bustout_xgb_accuracy ADD bustout1 int;")
con.execute("UPDATE bustout_xgb_accuracy SET bustout1=1 WHERE bustout = 'Y';")
con.execute("UPDATE bustout_xgb_accuracy SET bustout1=0 WHERE bustout = 'N';")
```

View the table

```
pd.set_option('display.max_colwidth', 80)
bustout_ac = DataFrame.from_query("select * FROM
TRNG_TDU_TD01.bustout_xgb_accuracy")
bustout_pda = bustout_ac.to_pandas()
bustout_pda
```

acct_no	bustout	prediction	bustout1
257	N	0	0
300	N	0	0
257	N	0	0
300	N	0	0
257	N	0	0

Bustout1 and prediction are the same format

Lab 8g: XGBoost – Accuracy

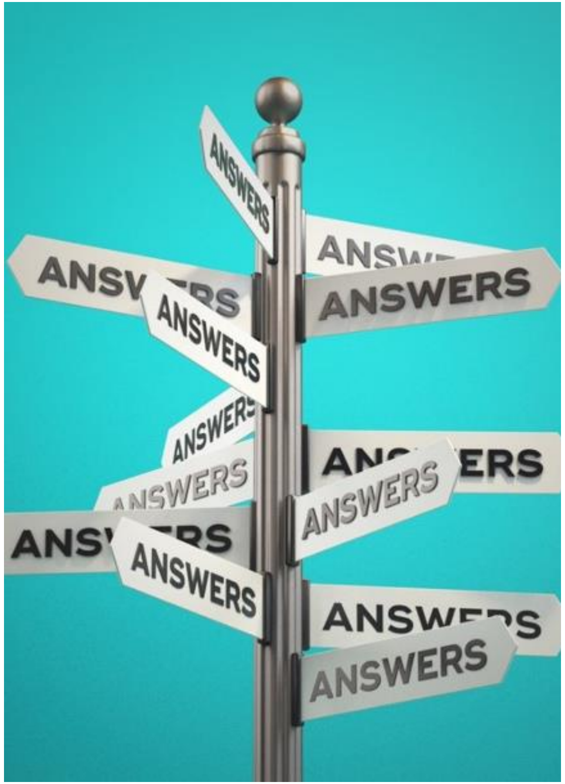
Check Accuracy

```
bustout_accr = DataFrame.from_query("SELECT (SELECT count(acct_no)*1.00
FROM bustout_xgb_accuracy \
WHERE bustout1 = prediction) / (SELECT count(acct_no) \
FROM bustout_xgb_accuracy) AS PA;")
bustout_paccr = bustout_accr.to_pandas()
bustout_paccr
```

	PA
0	0.92

Current Topic – XGBoost

- Predictive Analytics Overview
- Correlation
- Ensemble Functions
  - Decision Forest Trees (with ConfusionMatrix)
  - XGBoost
- **Summary**



Summary

In this module, you have learned how to:

- Calculate Correlation Matrix without moving data out from Vantage
- How to train Decision Trees Models in Vantage and in Python
- Build other advanced ensambled models tres like Xgboost

Thank you.