



7 Data Solutions

BOYD CORPORATION

SCORECARD DATABASE

Carlos Oyuela-Mora, Christopher Svenson, Qianwen Polly Zhang,
My Dinh, Yian Liou, John Shen, Kevin Khuu, Yitong Ellie Pan, Young Lin

BUSINESS OVERVIEW

Boyd Corporation. The company works as a supplier for other large companies such as Flextronics, Apple, and even Tesla. They specialize in engineered rubber, plastic, metal, and environmental sealing. Their headquarters is located in Modesto, California, but multiple locations exist around the United States such as Gaffney, South Carolina and Portland, Oregon.

Boyd Corp. approached Team 7 to design a database to facilitate their selection of suppliers. As this project was initiated by their Quality Department, a heavy emphasis was placed on Quality Assurance through the supplier selection process: the Supplier Scorecard - a simple display of how suppliers perform on four metrics (Customer Service, Price, On-time Delivery, and Quality). Critically, they needed to replace their current system of manually updated, disparate Excel spreadsheets. After designing the database to capture these metrics and all of their component parts in order to address the immediate needs of the Quality Assurance department at Boyd, Team 7 has gone on to create a much more comprehensive database for Boyd Corp, capturing sales data, shipping data, materials data, parts data, etc. touching all parts of their complex sales and manufacturing process.

RELATIONAL DESIGN

1. **Customer** (CID, Cname, Contact, Address, EID⁷)
2. **Purchase Order** (PID, Cname, Date, Drawing, PartID²⁸, MatID, Description, Quantity, Shipping Instruction, Unit price, Quality Clauses)
3. **Production Model** (PID², PartID²⁸, EID⁷, MatID⁹, Description, Quantity, Quote)
4. **Sales Order** (SID, PID², CID¹, PartID²⁸, Part Name, Quantity, Price, Request Date, Due Date, Notes)
5. **Work Order** (WID, SID⁴, PartID²⁸, MatID⁹, Work Order Description, Request Date, Schedule Date, Quantity, Hours)
6. **Work Order Details** (WID⁵, MID²⁶, FID¹⁹, EID⁷, Date, Sequence, Time In, Time Out)
7. **Employee** (EID, Supervisor ID Last, First, SSN, Department, Salary, Commission, Hourly Wage)
8. **Supplier:** (Supplier-ID, Company-name, Address, Contact, Title, Phone/Fax, Scope, ISO?(certificated?), Aerospace or ITAR)
9. **Material:** (MatID, Supplier-ID⁸, Material-name, Description, Expiration-date)
10. **Supplier Contact:** (EID⁷, Lname, MI, Fname, E-mail, Phone #)
11. **Scorecard:** (Supplier-ID⁸, FID¹⁹, Supplier-name, Aerospace/ Defense, Avg, 1st-half, 2nd-half, Quality, OTD, Price, CService_Ranking, SNCR-Qty, DPM (defective per million), OTD%, CAction#¹⁷, PAction#¹⁶)
12. **CNCR Log**(CNCR#, CID¹, SID⁴, PartID²⁸, Dated issued, Sales support representative, Reject quantity, Rejected quantity value, UOM (unit of measure), Date Forward for RCA/CA/PA, initial response, Customer Document(attachment), CodeID, RMA(Return material authorization)(1/0/n/a), RMA receipt date, QTY(Boyd examines), Qty RCVD(the quan accepted finally, Disposition, CAPA effect(1/0), Follow-up, Follow-Up/Completion/Verification DATE, Credit Date, REbill date, Date Closed)
13. **SNCR Log**(SNCR#, Date issued, Supplier_OID³⁰, Supplier-ID⁸, PID², PartID²⁷, Reject quantity, Rejected quantity value, UOM²⁷ (unit of measure), Area Discovered, Photo links(attachment), CodeID¹⁵, RMA(Return material authorization)(1/0/n/a), CAPA REQ'D, Debit Po, Purchasing Completed Action, Credit received, Notes, Supplier CAPA Response, Disposition, Follow-Up/Completion/Verification DATE, Date Closed)
14. **IPNCR** (IPNCR#, WID⁵, PartID²⁸, CodeID¹⁵, Work-cell, Dated issued, Rejected-QTY, UOM, Rej-QTY-value, Photo-links, champion(s), Initial-response-date, Capa-req'd, Completion-date, verification-date, Date-closed)
15. **Reject Code:** (CodeID, Year, Reject Reason, Number of reject Code)
16. **Preventive Action** (PAction#, Supplier-ID⁸, CID¹, CodeID¹⁵, Date, Quantity Value)
17. **Corrective Action** (CAction#, Supplier-ID⁸, CID¹, CodeID¹⁵, Date, Quantity Value)
18. **Facility_Produced_Parts**(FID¹⁹, PartID²⁸)
19. **Facility**(FID¹⁹, Location, Num_Employees, Operating_Days, Operating_Hours)
20. **Receive_shipments:** (FID¹⁹, SID⁴, Supplier_OID, Date, Time)
21. **Ship_to:** (FID¹⁹, CID, PID, SID, quantity, date)
22. **Facility_Manager**(FID¹⁹, EID⁷)

23. **Departments**(FID¹⁹, Department_Name)
24. **Machine_Makes_Part**(MID²⁵, FID¹⁹, M_Number_ID²⁵, PartID²⁸, Efficiency)
25. **Machine**(MID, FID¹⁹, M_Number_ID, Name, Years_of_Use, Notes)
26. **Inventory**:(PartID, Bin_n0, Quantity, Description, Supplier_OID, Order_date, Expiration_date)
27. **Part**(PartID, Name, Department, Acceptable, Manufactured, Manufactured_at_FID, Manufactured_by_MID, Obtained_from, Prototype, ManuDate, TestDate, TestedBy, UOM (Unit of Measurement))
28. **Uses_Material**(PartID²⁸, Mat_ID)
29. **Shipping_Capabilities**(FID¹⁹, Shipping_Bay_Sizes, Shipping_Bay_Numbers, Distance_to_Airport, Distance_to_Rail, Distance_to_Sea)
30. **Supplier_Order** : (Supplier_OID, Supplier-ID⁸, SID⁴, PartID²⁸, MatID⁹, Quantity, Notes, Purchased_Date, Due_Date, Price_Per_Unit, Total)

NORMALIZATION

Original Relation:

Scorecard (Supplier-ID⁸, FID¹⁹, Supplier-name, Aerospace/ Defense, Avg, 1st-half, 2nd-half, Quality, OTD, Price, CService_Ranking, SNCR-Qty, DPM (defective per million), OTD%, CAction#¹⁷, PAction#¹⁶)

Functional Dependencies:

- {Supplier-ID⁸, FID¹⁹} → {Supplier-name, Aerospace/ Defense, Avg, 1st-half, 2nd-half, Quality, OTD, Price, CService_Ranking, SNCR-Qty, DPM (defective per million), OTD%, CAction#¹⁷, PAction#¹⁶}
- {Supplier-ID⁸} → {Supplier-name, Aerospace/ Defense}
- {1st-half, 2nd-half} → {Avg}
- {Quality, OTD, Price, CService_Ranking, SNCR-Qty} → {1st-half, 2nd-half}
- {OTD%} → {OTD(Score)}
- {SNCR-Qty, DPM} → {Quality(Score)}

We used our scorecard relation to do our normalization analysis. The last two attributes corrective and preventive action are our future implementation to improve our scorecard database, so let's set aside those two in our analysis. Starting from our scorecard's criteria attributes, quality, on time delivery, price, customer service ranking, supplier non-conformance report, defectives, and OTD%, all of which are multivalued attributes. Each has a score value for the first half of the year and another score for the second half of the year. Because 1NF can only have single valued attribute, we remove all of those multivalued attributes from the scorecard relation and normalized into 1NF as below.

First Normal Form:

- **Supplier_name**(Supplier-ID⁸, Supplier_name)
- **Supplier_type**(Supplier-ID⁸, Aerospace/Defense)
- **Scorecard** (Supplier-ID⁸, FID¹⁹, Supplier-name, Aerospace/ Defense, Avg, 1st-half, 2nd-half)
- **Quality** (Supplier-ID⁸, FID¹⁹, Quality1, Quality2)
- **OTD** (Supplier-ID⁸, FID¹⁹, OTD1, OTD2)
- **Price** (Supplier-ID⁸, FID¹⁹, Price1, Price2)
- **CService_Ranking** (Supplier-ID⁸, FID¹⁹, CService_Ranking1, CService_Ranking2)
- **SNCR-Qty** (Supplier-ID⁸, FID¹⁹, SNCR-Qty1, SNCR-Qty2)
- **DPM** (defective per million) (Supplier-ID⁸, FID¹⁹, DPM1, DPM2)
- **OTD%** (Supplier-ID⁸, FID¹⁹, OTD%1, OTD%2)

We created new relations for each of the multivalued attributes. For example, quality has the composite key, Supplier-ID and FID, from the original scorecard relation, and then we split it into quality1 and quality 2 to record the two scores for each half of the year. The same relationship is applied to the rest of the multivalued attributes. Now, the attributes Supplier-name and Aerospace/Defense in the scorecard relation violate the 2NF condition because as non-prime

attributes, they can be determined from the subset of the composite key, Supplier-ID. Therefore, we can further normalize the scorecard relation into 2NF as below.

Second Normal Form:

- Supplier_name(Supplier-ID⁸, Supplier_name)
- Supplier_type(Supplier-ID⁸, Aerospace/Defense)

Two new relations are created in 2NF: Supplier-name and Supplier_type. They are full dependencies relations with Supplier-ID as the primary key that directly determines the supplier name or the type, Aerospace/Defense.

Third Normal Form:

- Supplier_Avg Score(1st-half, 2nd-half, Avg)
- Supplier_Semi-Annual Score(OTD, Price, CService_Ranking, SNCR-Qty, 1st-half, 2nd-half)
- Supplier_OTD Ranking(OTD%, OTDScore)
- Supplier_Quality(DPM (defective per million), SNCR-Qty, QualityScore)

Four new relations are created in 3NF: Supplier_AvgScore, Supplier_Semi-Annual Score, Supplier_OTD Ranking, and Supplier_Quality. Since R is in 2NF, we now need to eliminate all of the non-prime attributes that are transitively dependent on the CK. Supplier_Avg Score is a composite of the 1st and 2nd half scores. The Semi-Annual score for each 6 month period can be determined by taking the average of the four metrics given, OTD, Price, Cservice Ranking, and SNCR-Qty. The OTD ranking can be determined by which threshold that a suppliers OTD% satisfies. Supplier_Quality is a joint measure of both the defects per million materials, and how many NCR errors they received in either 6 month period.

Boyce-Codd Normal Form:

After converting adding new relations in 3NF, this is also in BCNF as each of these four new relations is a trivial functional dependency and the rest of the relations have super keys for schema R.

Original Relation:

Machine(MID, FID¹⁹, M_Number_ID, Name, Years_of_Use, Notes)

Functional Dependencies:

- {MID} → {Name}
- {MID, FID, M_Number_ID} → {Years_of_Use, Notes}

First Normal Form:

Currently not in First Normal Form because Notes is a multi-valued attribute. Change to First Normal Form by creating a relation specifically for machine notes.

- Machine(MID, FID¹⁹, M_Number_ID, Name, Years_of_Use)
- Machine_Notes(MID, FID¹⁹, M_Number_ID, Note)

Second Normal Form:

Currently not in Second Normal Form because MID, FID, M_Number_ID → Name is a partial attribute (only MID → Name). Change to Second Normal Form by creating a relation specifically with just MID and Name.

- Machine_Names(MID, Name)
- Machine(MID, FID¹⁹, M_Number_ID, Years_of_Use)
- Machine_Notes

Already in Third and Boyce-Codd Normal Form because no non-prime attributes are transitively dependent, and for all functional dependencies $X \rightarrow Y$, X is a superkey.

Original Relation:

SNCR Log(SNCR#, Dated issue, Supplier_OID³⁰, Supplier-ID⁸, PID², PartID²⁷, Reject quantity, Rejected quantity value, UOM²⁷ (unit of measure), Area Discovered, Photo links(attachment), CodeID¹⁵, RMA(Return material authorization)(1/0/n/a), CAPA REQ'D, Debit Po, Purchasing Completed Action, Credit received, Notes, Supplier CAPA Response, Disposition, Follow-Up/Completion/Verification DATE, Date Closed)

Functional Dependencies:

- {SNCR#} → {Date issued, Supplier_OID, PID, Reject quantity, Area discovered, Photo links, CodeID, RMA(Return material authorization)(1/0/n/a), CAPA REQ'D, Debit Po, Purchasing Completed Action, Credit received, Notes, Supplier CAPA Response, Disposition, Follow-Up/Completion/Verification DATE, Date Closed}
- {Supplier_OID} → {Supplier-ID}
- {PartID} → {UOM}
- {PID, Reject quantity} → {Rejected quantity value}
- {PID} → {PartID}
- {Debit PO} → {Credit Received}

First Normal Form:

Not in First Normal Form because Photo links and Notes are multi-valued attributes. Change to First Normal Form by creating a relation specifically relating photo links to SNCR# and notes to SNCR#.

- SNCR Log(SNCR#, Dated issue, Supplier_OID³⁰, Supplier-ID⁸, PID², PartID²⁸, Reject quantity, Rejected quantity value, UOM (unit of measure), Area Discovered, CodeID¹⁵, RMA(Return material authorization)(1/0/n/a), CAPA REQ'D, Debit Po, Purchasing Completed Action, Credit received, Supplier CAPA Response, Disposition, Follow-Up/Completion/Verification DATE, Date Closed)
- SNCR_Notes(SNCR#, Note)
- SNCR_Photos(SNCR#, Photo link)

Second Normal Form:

Already in Second Normal Form because there are no partial dependencies.

Third Normal Form:

Not in Third Normal Form because there are transitive dependencies: SNCR# → Supplier_OID, PID, Reject Quantity; Supplier_OID → Supplier-ID; PID, Reject quantity → Rejected quantity value; PID → PartID; PartID → UOM. This information is repeated in the tables Supplier_Order, Purchase_Order, and Part.

- SNCR Log(SNCR#, Dated issue, Supplier_OID³⁰, PID², Reject quantity, Area Discovered, CodeID¹⁵, RMA(Return material authorization)(1/0/n/a), CAPA REQ'D, Debit Po, Purchasing Completed Action, Credit received, Supplier CAPA Response, Disposition, Follow-Up/Completion/Verification DATE, Date Closed)
- Supplier_Order
- Purchase_Order
- Part

Boyce-Codd Normal Form:

Already in Boyce-Codd Normal Form because for all functional dependencies $X \rightarrow Y$, X is a superkey.

Original Relation:

Supplier: (Supplier-ID, Company-name, Address, Contact, Title, Phone/Fax, Scope, ISO?(certificated?), Aerospace or ITAR)

Functional Dependencies:

- {Supplier-ID} → {Company-name, Address, Contact, ISO, Aerospace or ITAR}
- {Supplier-ID, Contact} → {Address, Title, Phone/Fax}

First Normal Form:

Phone/Fax can hold more than one number, thus it must be reduced to achieve 1NF.

- Supplier'(Supplier-ID, Company-name, Address, Contact, Title, Scope, ISO_certified, Aerospace or ITAR)
- Phone_Fax(Supplier-ID, Number, Phone or Fax)

Second Normal Form:

The relation is now in 2NF as the Candidate Key is single valued (i.e. no non-empty proper subset exists for the CK)

Third Normal Form:

The relation is in 3NF as all attributes are directly dependent on the CK

Boyce-Codd Normal Form:

- Supplier''(Supplier-ID)
- Company-name(Supplier-ID, Company-name)
- Company-address(Supplier-ID, Address)
- Company-contact(Supplier-ID, Contact)
- Contact-title(Supplier-ID, title)
- Company-certs(Supplier-ID, ISO-certified, Aerospace_or_ITAR)
- Phone_Fax(Supplier-ID, Number, Phone or Fax)

Original Relation:

Work Order (WID, SID⁴, PartID²⁸, MatID⁹, Work Order Description, Request Date, Schedule Date, Quantity, Hours)

Functional Dependencies:

- {WID, PartID, Quantity, Schedule Date} → {Work Order Description}
- {PartID, Quantity} → {Hours, Schedule Date}
- {SID} → {Request Date, PartID, Quantity}
- {PartID} → {MatID}

First Normal Form:

A Sales Order can consist of multiple parts, each of which can be made with multiple materials, has a (possibly unique) work order description, hours to complete, quantity, and schedule date.

- Work Order'(WID, SID, Request Date)
- Part(SID, PartID, Work Order Description, Quantity, Hours, Schedule Date)
- Material(PartID, MatID)

Second Normal Form:

- Work Order'(WID, SID, Request Date)
- Part(SID, PartID, Quantity, Hours, Schedule Date)
- WorkDescription(PartID, Work Order Description)
- Material(PartID, MatID)

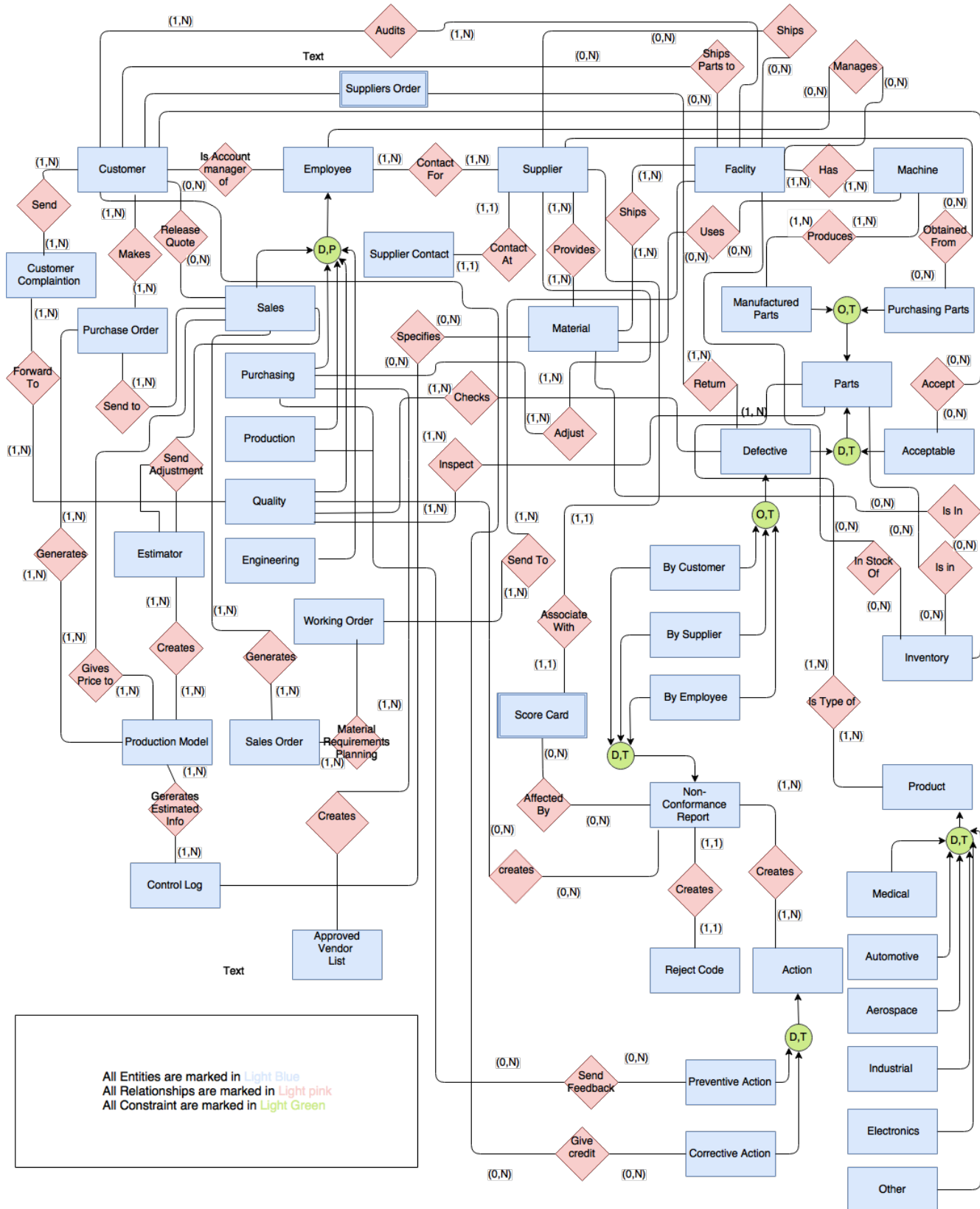
Third Normal Form:

- Work Order'(WID, SID)
- Request(SID, Request Date)
- Part(SID, PartID, Quantity, Schedule Date)
- Hours(PartID, Quantity)
- WorkDescription(PartID, Work Order Description)
- Material(PartID, MatID)

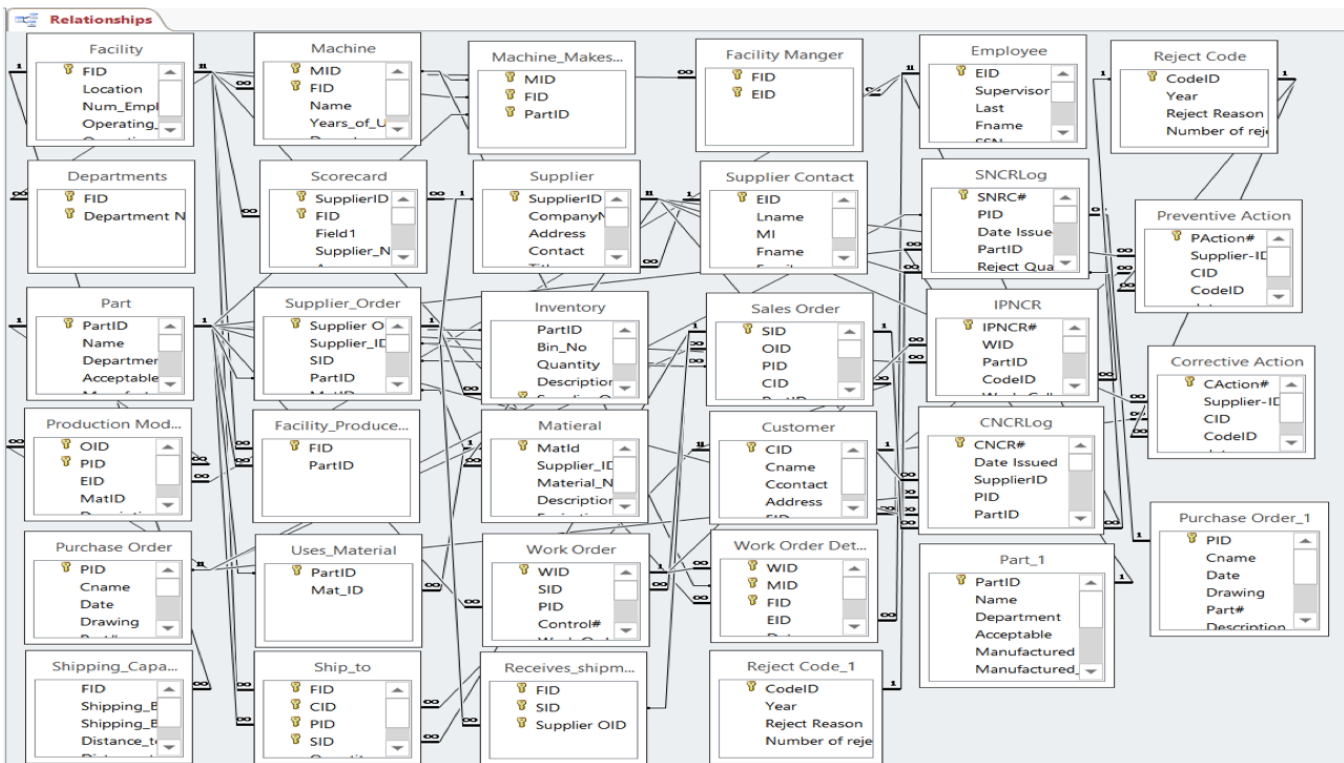
Boyce-Codd Normal Form:

Already in Boyce-Codd Normal Form.

SIMPLIFIED EER



ACCESS RELATIONSHIP VIEW



ACCESS FORM VIEW

The screenshot shows the Access Form View for the CNCRLog table. The form contains the following fields and values:

CNCR#		Notes	Damage
Date Issued	3/2/2016	Suppliers CAPA	Shipper responsible
SupplierID	1	Disposition	Open
PID	10	Follow-Up/Com	
PartID	1	Date Closed	
Reject Quantity	54	CID	1
Rejected Quanti	\$18,874.00	SID	1
Unit Of Measure	cm	Sales Support R	9
Area Discovrec			
Photo Links			
CodeID	44		
Return Material	Y		
CAPA Req'd			
Debit Po			
Purchasing Com	Y		
Credit Received	7/28/2016		

At the bottom, the record navigation bar shows: Record: 1 of 11, No Filter, Search.

The CNCRLog Form serves to allow the user to completely enter a CNCR (Customer Non-Compliance Report). This is an important part of Boyd's quality assurance process as it collects data about customer dissatisfaction with provided units. This allows Boyd to investigate the causes of their dissatisfaction and ascertain whether it was a materials-based issue (which the Supplier would need to address) or a manufacturing-based issue (which Boyd would need to address, or whether another cause has created the issue at hand).

SNCRLog

SNCR#	<input type="text" value="1"/>	Notes	<input type="text" value="Issue with adhesion factor"/>
PID	<input type="text" value="2"/>	Suppliers CAPA	<input type="text" value="Pending"/>
Date Issued	<input type="text" value="1/3/2016"/>	Disposition	<input type="text" value="Pending"/>
PartID	<input type="text" value="1"/>	Follow-Up/Com	<input type="text" value="11/24/2016"/>
Reject Quantity	<input type="text" value="58"/>	Date Closed	<input type="text"/>
Rejected Quanti	<input type="text" value="\$2,595.00"/>	SupplierID	<input type="text" value="1"/>
Unit Of Measure	<input type="text" value="cm"/>		
Area Discoverec	<input type="text"/>		
Photo Links	<input type="text"/>		
CodeID	<input type="text" value="12"/>		
Return Material	<input type="text" value="Y"/>		
CAPA Req'd	<input type="text" value="Y"/>		
Debit Po	<input type="text"/>		
Purchasing Corr	<input type="text"/>		
Credit Received	<input type="text" value="\$151.35"/>		

Record: 1 of 10 No Filter Search

The SNCRLog Form serves to allow the user to completely enter a SNCR (Supplier Non-Compliance Report). This is another important part of Boyd's quality assurance process as it collects data about issues Boyd catches with Supplier-provided materials or fabrications. This allows Boyd to catalog issues with Suppliers, detail issues, to document steps taken to address issues, and provide explanations to customers as needed.

Supplier_Order

Supplier OID	<input type="text" value="3"/>
Supplier_ID	<input type="text" value="19"/>
SID	<input type="text" value="2436"/>
PartID	<input type="text" value="115"/>
MatID	<input type="text" value="12"/>
Quantity	<input type="text" value="1000"/>
Notes	<input type="text" value="Express Shipping"/>
Due_Date	<input type="text" value="12/13/2016"/>
Price_Per_Unit	<input type="text" value="3"/>
Total	<input type="text" value="3000"/>

Record: 1 of 11 No Filter Search

The Supplier_Order Form serves to allow the user to completely enter an order that has been placed to a supplier. This simple form captures materials, quantities, shipping instructions and other notes, due dates, price per unit, and quantity. Things like units of material are encoded in the Materials Table and dependent on the MatID number, thus eliminating the need to enter this information in the Supplier_Order Form. Similarly, all contact information on the Supplier side is dependent on the Supplier_ID and Boyd Sales contacts are captured via the SID (Sales ID). The goal of this form is to simplify the order documentation process.

ACCESS REPORT VIEW

SNCRLog

PID	SNRC#	Date Issued	PartID	t Quantity	ected Quantity	Value	CodeID
2	1	1/3/2016	1	58		\$2,595.00	12
3	4	8/20/2016	6	89		\$496.00	12
5	9	8/17/2016	1	88		\$1,949.00	55
6	5	3/31/2016	7	50		\$2,235.00	3
	10	6/1/2016	2	59		\$3,782.00	44
	6	1/8/2016	8	17		\$3,989.00	33
8	7	1/3/2016	9	77		\$1,918.00	7
9	8	1/19/2016	10	22		\$4,107.00	6
10	2	1/25/2016	3	20		\$3,661.00	1
11	3	7/8/2016	4	49		\$1,444.00	5

The SNCRLog Report displays all of the SNCRLog details. As this is a critical metric, a comprehensive report is important for the Quality Assurance department, the Bidding Department, and the Sales Department. The report can of course be refined by any number of attribute values/ranges.

CNCRLog1

SupplierID	CNCR#	Date Issued	PID	PartID
1	11			
2	1	3/2/2016	10	1
7	9	11/11/2016	8	4
	2	3/23/2016	4	4
10	3	1/10/2016	5	3
	10	1/18/2016	7	6
11	7	9/13/2016	8	9
	4	7/9/2016	6	5
13	8	11/19/2016	6	2
14	5	3/16/2016	7	7
15	6	9/16/2016	10	8

The CNCRLog Report displays all of the CNCRLog details. This, too, is a critical metric which requires a comprehensive report. As above, it is important to many of Boyd's departments and NCR in general are considered important within Boyd as Boyd's reputation hinges on their reliability (which partially hinges on that of their Suppliers and materials).

QUERIES

Query 1: Optimization of Production Cost for Parts

Justification

When an order is taken from a customer to produce a part, an optimized production schedule for the machines should be generated so that Boyd can produce enough while minimizing the cost for production. This machine scheduling is designed to ensure the production is efficiently covered and that the operational production cost is minimized.

Implementation

Assume the available hours for each machine are known. In this example, assume an order necessitates the fabrication of 10,000 units of part 12 in 5 days. SQL query is run to check which machines are capable of making part 12 and to obtain the quantity of part 12 each machine can produce per unit time. And then we can input all the data to AMPL to get an optimized solution.

1. SQL

In order to get all the information about the machines that are able to produce part12. We need to retrieve the MachineID, efficiency and operational cost. The code for SQL is shown below:

Machine

```
Select MID, Efficiency, OperationalCost
From Machine_Makes_Part
Where PartID=12 and FID=1;
```

MID	Efficiency
3	88
7	120
13	146

Figure: Sample SQL

2. AMPL

Using data generated by the SQL query, a linear programming model is constructed to solve this task assignment problem. The next figure summarizes this program and is explained in detail below:

c_i – total cost for machine i to produce every hour

P_i – Operational cost for machine i

w_i – Wage for the employee to operate on machine i

y_{ij} – The number of time allocated for the machine i to produce on day j

z_{ij} – The number of available time for machine i to produce the part each day

d - the available day for production before due day

M - the total quantity to produce

Objective: Minimize the operational cost

$$\min \sum_{j=0}^d \sum_{i=1}^n c_i * y_{ij}$$

Subject to:

1. Production constraint

$$\sum_{j=0}^d \sum_{i=1}^n x_i * y_{ij} \geq M$$

2. Time constraint

$$y_{ij} \leq z_{ij} \forall ij$$

$$c_i = p_i + w_i \forall i$$

Figure: Linear Programming Model Formulation

The decision variable is Y_{ij} , denoting the number of hours assigned to each machine per day. There are four parameters, which can be obtained from the table generated by the SQL query. The objective function minimizes the total cost: the summation of operational costs of each machine multiplied by the machine hours used. The first constraint ensures sufficient production to fulfill the order and the second constraint ensures that each machine is not scheduled to produce more than its available hours per day. The last constraint is a formula for the operational cost per hour.

After the formulation, the data and the linear programming model will be saved as a .dat file and a .mod file respectively and then solved using CPLEX in AMPL. In this particular example, there are 3 machines and 5 days to

```
#####
#Assumption: we have three machine available to make part 12
#We have 5 days to make the part
#The available hours and operational cost are by assumption
#####

var y{i in 1..3, j in 1..5} >= 0;
# total cost for machine i per hour
var c{i in 1..3};
# Operational cost for each machine

param x{i in 1..3}; # the efficiency of each machine
param p{i in 1..3}; # operational cost for machine i per hour
param w{i in 1..3}; # wage per hour to control machine i
param z{i in 1..4, j in 1..5};
# number of hours available for machine i to produce on day j
param M; # the total quantity to produce

minimize totalCost:
    sum{i in 1..3, j in 1..5} c[i] * y[i,j];

subject to production: sum{i in 1..3, j in 1..5} x[i] * y[i,j] >= M;
subject to time{i in 1..3, j in 1..5}: y[i,j] <= z[i,j];
subject to cost{i in 1..3}: c[i] = p[i] + w[i];
```

Figure: The code in AMPL

```
# Total quan to produce
param M := 10000;
# The efficiency of each machine
param x := 1 88 2 120 3 146;
# Operational cost per hour
param p := 1 20 2 26 3 30;
# Wage of technicians per hour
param w := 1 15 2 15 3 18;
# Available hours for each machine
# per day
param z:  1  2  3  4  5:=
1  5  6  8  8  6
2  6  6  5  7  8
3  5  6  5  4  6;
```

Figure: .dat file for this example

schedule.

After inputting the model and data into AMPL, the final result is given as follows. The AMPL output shows how many hours each machine should spend on each day. The first column of the output indicates the machine number and the second column indicates the day. For example, row 1 says that machine 1 should produce 5 hours in day1. Based on our data, the minimized operational cost is \$3500.23.

```
ampl: display totalCost;
totalCost = 3500.23

ampl: display y;
y :=
1 1 5
1 2 6
1 3 1.86364
1 4 8
1 5 6
2 1 6
2 2 6
2 3 5
2 4 7
2 5 8
3 1 5
3 2 6
3 3 5
3 4 4
3 5 6
;
```

Figure: Output for Query 1

Query 2: Purchase Quality Optimization

Justification

This query will calculate the right amount of material to buy from a specific supplier to produce a part by calculating the 95% Confidence Interval for the defective rate of that supplier. With this predicted defective rate, Boyd can achieve 95% confidence that they have enough material for production.

Implementation

Assume the defective rate of each batch of material provided by supplier is Independently Distributed and the Central Limit Theorem holds. Also,

As an example, assume the need to purchase MatID 99 to produce the part for production order PID 30.

1. SQL

Generate two tables. The first shows the required material and its supplier, which is used to find the defective rate (%) in the second table by using all the historical data for the subject supplier.

Create table *Parts* as

Select m.MatID, po.Quantity * pm.Quantity, m.Supplier-ID

From Production Order as po, Production Model as pm, Material as m

Where po.PID=30 and pm.MatID=99 and pm.PID=po.PID and m.MatID=pm.MatID;

Create table *Supplier_performance* as

Select SNCR.supplier-ID, SNCR.Reject quantity/p.quantity as defective_rate

From SNCR Log as SNCR, Purchase Order as p, PARTS as pa

Where SNCR. Supplier-ID=pa. Supplier-ID and SNCR.PID=p.PID;

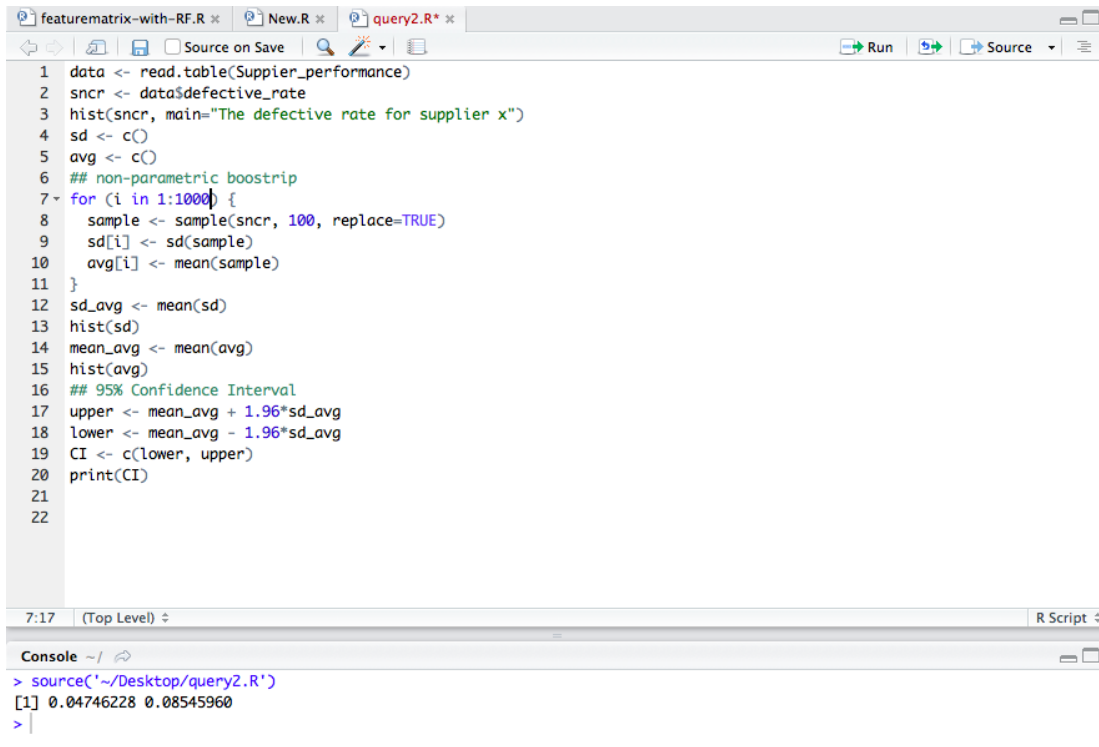
DefectRatio	SupplierID	Supplier_ID
0.0004	13	13

Figure: Sample SQL Output

2. R

After creating the *Supplier_performance* table for the historical defective rates from the supplier, all the defective rate data is pulled into R. By the Central Limit Theorem, the mean percentage will be normally distributed. Using nonparametric bootstrap, the mean and standard deviation are calculated in one thousand iterations. These results are then used to compute the 95% confidence interval for the percentage of defective parts and determine quantity of materials to be purchased from the supplier. By taking the upper limit of this confidence interval, Boyd can be reasonably sure it has enough material to produce the parts for a given order. Determining this optimal order quantity will generate cost-savings and improve bid offers.

In the example, Boyd's data shows the defective rate for this specified supplier(NK Corporation) is approximately 7%. Based on the calculated 95% confidence interval, the true mean is contained inside the interval. Accordingly, Boyd should use the upper limit of 0.0855 to determine the optimal purchasing quality.



```

1 data <- read.table(Supplier_performance)
2 snrcr <- data$defective_rate
3 hist(snrcr, main="The defective rate for supplier x")
4 sd <- c()
5 avg <- c()
6 ## non-parametric bootstrap
7 for (i in 1:1000) {
8   sample <- sample(snrcr, 100, replace=TRUE)
9   sd[i] <- sd(sample)
10  avg[i] <- mean(sample)
11 }
12 sd_avg <- mean(sd)
13 hist(sd)
14 mean_avg <- mean(avg)
15 hist(avg)
16 ## 95% Confidence Interval
17 upper <- mean_avg + 1.96*sd_avg
18 lower <- mean_avg - 1.96*sd_avg
19 CI <- c(lower, upper)
20 print(CI)
21
22

```

7:17 (Top Level) R Script

```

> source('~\\Desktop\\query2.R')
[1] 0.04746228 0.08545960
>

```

Figure: Code and Output for Query 2

Query 3: Supplier Selection

Justification

When Boyd receives a new order request from the customer, they need to determine the materials used and the supplier for the materials to create a production model and give the customer the quote. In query 3, if the supplier chosen matches the order's characteristics and customer's preference, Boyd will be less likely to provide extra production models that would incur extra expenses on Boyd. The model can also decrease the response time between the order request and price quote, thus increase the chance of getting new orders and creating Boyd's. Sales orders data are used to train the model, which contains the details about the customer orders. The test set will be based on the supplier order that was chosen for those customer orders. By using data from the past, the random forest model can pick up the pattern of the order's characteristics, the customer's preference and the supplier's capacity to fulfill the order.

Implementation

1. Extract data from Access using SQL:

```

SELECT po.MatID, so.Quantity, so.Quantity, so.Shipping_Instruction, po.SupplierID
FROM Suppliers_Order as so, Purchase_Order as po, Sale_Order as sa,
WHERE sa.PID = po.PID AND sa.PartID = po.PardID AND sa.SID = so.SID

```

	MatID	Quantity	Shipping_Instruction	Unit_Price	CustomerID	SupplierID
0	BYC104	4	9	26	3	A
1	BYC104	4	9	22	2	B
2	BYC104	6	10	30	1	C

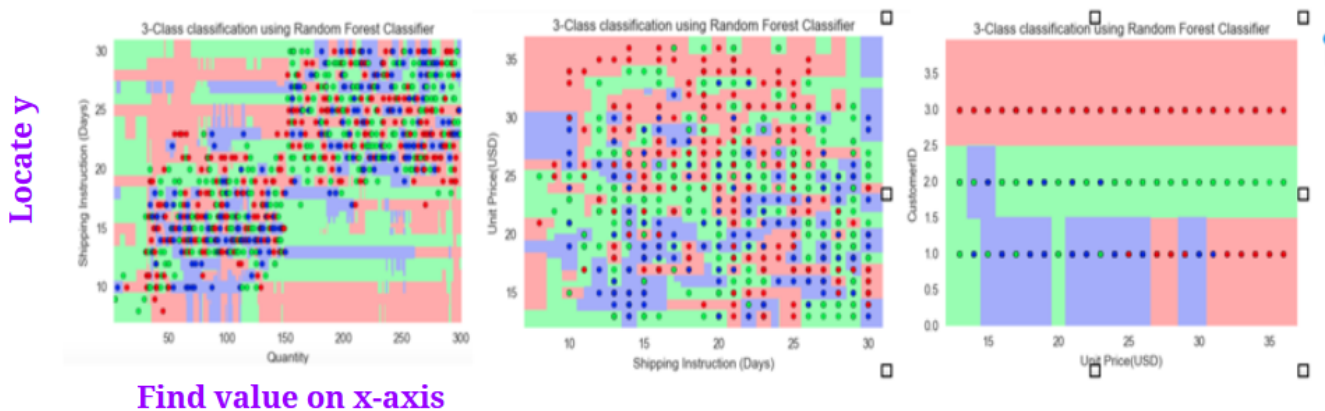
Figure: Sample SQL Output

2. Build model and train it with 80% of the dataset.

The package from scikitlearn library is used to build the random forest classification model. It uses 53 decision trees, which is carefully chosen by cross-validation method to check which parameter gives the highest accuracy.

Using one hot encoder for categorical values, our model is built on 9 dimensions. Each plot below is a 2D visualization of one dimension of the model, where x and y axis corresponds to two features of the orders. Suppliers will have their own region marked by unique color. The model assigns supplier to a new order if the location of the order is closed to the region of that specific seller. In other words, the recommended supplier is generated by matching features of the order to the suppliers.

The dot should now have its region's color!



Python code to build model:

```
In [82]: from pandas import DataFrame, Series
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.cross_validation import train_test_split
from sklearn.metrics import log_loss

In [83]: df = pd.read_csv("purchase.csv")
df_only = pd.get_dummies(df, columns = ['MatID'], drop_first = True)
target = df_only['SupplierID']
purchase = df_only.drop(['SupplierID'], 1)
import seaborn as sns
%matplotlib inline
sns.pairplot(df, kind = 'scatter', diag_kind='kde')
X_train, X_test, y_train, y_test = train_test_split(purchase, target, test_size = 0.2)
#CV to pick parameter
from sklearn.cross_validation import cross_val_score
def cv_stats(cv_score):
    """ Returns the mean and standard deviation in a readable format"""
    mean = np.mean(cv_score)
    std = np.std(cv_score)
    return mean, std
k = 40
while k < 80:
    clf = RandomForestClassifier(k)
    score = cross_val_score(clf, X_train, y_train, cv=5)
    stats = cv_stats(score)
    print("For k={0}, k, "; Average={0}, STDEV={1}".format(*stats))

#standardize data
mean = X_train.mean(axis=0)
std = X_train.std(axis=0)
X = (X_train - mean) / std
clf = RandomForestClassifier(n_estimators=53, random_state=123)
clf.fit(X_train, y_train)
preds = clf.predict_proba(X_test)
preds2 = clf.predict(X_test)
print(log_loss(y_test, preds))

from matplotlib.colors import ListedColormap
cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])
plot_step = 0.02 # fine step width for decision surface contours
# Train
X = df.iloc[:, 1:3]
y = target
clf = RandomForestClassifier(n_estimators=35, random_state=123)
clf.fit(X, y)
scores = clf.score(X, y)
import pylab as pl
x_min, x_max = X.iloc[:, 0].min() - 1, X.iloc[:, 0].max() + 1
y_min, y_max = X.iloc[:, 1].min() - 1, X.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step), np.arange(y_min, y_max, plot_step))
# Plot either a single DecisionTreeClassifier or alpha blend the
# decision surfaces of the ensemble of classifiers
conv = lambda x : ord(x) - 65
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
numZ = np.array([conv(z) for z in Z])
numy = np.array([conv(a) for a in y])
numZ = numZ.reshape(xx.shape)
cs = plt.contourf(xx, yy, numZ, cmap=cmap_light)
plt.scatter(X.iloc[:, 0], X.iloc[:, 1], c=numy, cmap=cmap_bold)
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("3-Class Classification using Random Forest Classifier")
plt.xlabel('Quantity')
plt.ylabel('Shipping Instruction (Days)')
plt.show()
```


3. Test the model with 20% of the dataset.

$$\text{logloss} = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^M y_{ij} \log(p_{ij})$$

N: Number of records

M: Number of classes (features)

y_{ij} is a binary variable indicating whether class j was correct for example i

p_{ij} the model probability of assigning label j to instance i

The model's performance on the dataset is validated using the multi class logarithmic loss metric. The log loss function will measure the performance by penalizing the false classification on the test set. Our model log loss is 0.11 in which 0 means correctly classify all inputs.

Query 4: Inventory Cost Minimization

Justification

If Boyd knows the material and the quantity needed along with the selected supplier, Boyd can order those materials before peak-seasons when prices will be at their highest. By ordering in advance, Boyd can avoid the extra cost related to price surges during peak seasons and thus keep company expenditures to a minimum. This analysis can come after the process of selecting supplier from Query 3.

By training the model with the past supplier order dataset, the model can detect the pricing pattern (if any exists).

Implementation

1. Extract data from Access using SQL

```
SELECT .MatID, Quantity, SupplierID, Quantity, Purchase_Date
FROM Suppliers_Order
WHERE s.Date = sp.Date, so.Supplier_OID = sp.Supplier_OID
```

	MatID	Quantity	Purchase_Date	Supplier ID	Unit Cost
0	TFC48	114	1/2/06	Supplier B	28.08
1	TCC17	43	1/3/06	Supplier B	10.30
2	NTCM123	135	1/4/06	Supplier G	13.35

Figure: Sample SQL Output

2. Build the model and train it with 80% of the data set.
Scikitlearn library is used to build the model.

```
from pandas import DataFrame, Series
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.ensemble import RandomForestRegressor
from sklearn.cross_validation import train_test_split
```

```
inventory = pd.read_csv("inventory_order.csv")
inventory['month'] = pd.DatetimeIndex(inventory['Purchase_Date']).month
inventory['day'] = pd.DatetimeIndex(inventory['Purchase_Date']).day
inventory = inventory.drop(['Purchase_Date'], axis=1)
inventory_only = pd.get_dummies(inventory, columns=['MatID', 'Supplier ID'], drop_first=True)
#split data
local_train, local_test = train_test_split(inventory_only, test_size=0.2, random_state=123)
local_train.shape
local_test.shape
local_train_y = local_train["Unit Cost"]
local_train_x = local_train.drop(["Unit Cost"], axis=1)
local_test_y = local_test["Unit Cost"]
local_test_x = local_test.drop(["Unit Cost"], axis=1)
#build model
clf = RandomForestRegressor(n_estimators=230)
clf.fit(local_train_x, local_train_y)
preds = clf.predict(local_test_x)
```

```
#visualization:
import pydotplus
from sklearn import tree
from IPython.core.display import Image

clf = tree.DecisionTreeRegressor()
clf = clf.fit(local_train_x, local_train_y)
dot_data = tree.export_graphviz(clf, out_file='treevisual.dot')
import pydotplus
import os
dot_data1 = tree.export_graphviz(clf, out_file=None)
graph = pydotplus.graph_from_dot_data(dot_data1)
image = graph.write_png("inventory.png")
```

Utilizing the graphviz tool to generate a visualization of one decision tree in the model, check whether the random forest is an appropriate model for this dataset.

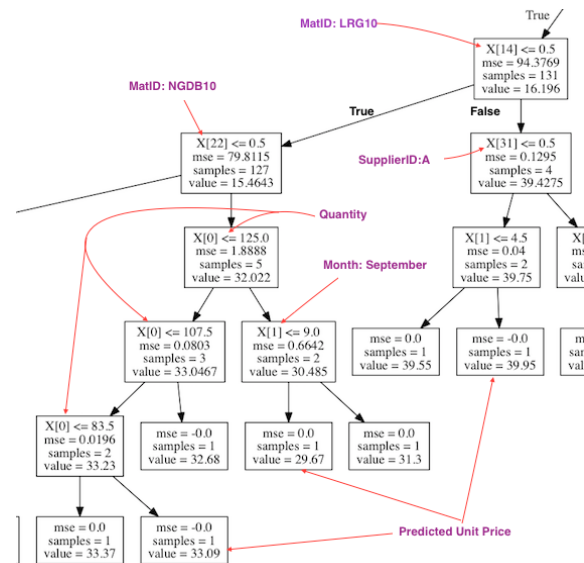


Figure: Decision Tree Section

3. Test the model with 20% of the data:

Mean Absolute Error metric is utilized to validate the performance of our model. Our average sum of the errors between the actual price and the predicted price is 0.7 over 700 inputs.

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| = \frac{1}{n} \sum_{i=1}^n |e_i|$$

$$AE = |e_i| = |y_i - \hat{y}_i|$$

$$Actual = y_i$$

$$Predicted = \hat{y}_i$$

Figure: Absolute Mean Error Formula

Query 5: Determine which department of Boyd has had the most occurrences of errors

Justification

Every half a year, every Boyd department must determine whether they have met their goals for the period, and how much they should adjust them by for the upcoming periods. By standardizing the mistakes and problems that occur in Boyd, the departments are able to focus on the root cause of those issues and prevent the errors from happening again. There is no straightforward solution to this as Boyd, must create attainable goals without oversimplifying any process. This query allows for deeper analysis into the root cause of issues by narrowing focusing on a specific standard problem and then finding the frequency that it occurs in a department.

Implementation

1. SQL: In order to help Boyd in the most effective way, the issues must be divided by their priority initially. Top priority errors have a high rejected quantity and high rejected quantity value percentage. The SQL code pulls these errors out of all Non-conformance reports by returning the instances of these errors which have surpassed a >15 percent threshold in both categories.

```
select count(s.SNCR#), s.CodeID
from SNCRLog as s, Purchase Order as p
where (s.rejectqty/p.qty) > .15
and (s.rejectqtyval/p.totalsalevalue) > .15
and p.pid = s.pid
```

```
select count(ip.IPNCR#), ip.CodeID
from IPNCR as ip, Work Order as w
Sales order as so, Purchase Order as p
where (ip.rejectqty/p.qty) > .15
and (ip.rejectqtyval/p.totalsalevalue) > .15
and p.pid = so.pid
and so.sid = w.sid
and w.wid = ip.wid
```

```
select count(c.CNCR#), c.CodeID
from CNCR as c, sales order as so,
Purchase Order as p
where (c.rejectqty/p.qty) > .15
and (c.rejectqtyval/p.totalsalevalue) > .15
and p.pid = so.pid
and so.sid = c.sid
```

2. Python: In order to find the frequency percentage of each department, a count is taken of the number of times each department has committed a high priority error. This is then divided by the total number of high priority errors. $f_i = f/n$. Since these errors have been standardized, they have been recorded and transformed into Reject Defect Codes. Each Non-Conformance Report has the appropriate Reject Defect Code attached to it. The Reject Defect Codes all start with a letter, and each letter works as a shortcut guide to pinpoint in which department the error came from.

F03	WRONG MATERIAL SHIPPED	VENDOR
F04	BAD WORKMANSHIP	VENDOR
F05	SHIPPED OVER/UNDER	VENDOR
F06	MARKING-WRONG #, INFO OMITTED	VENDOR
F07	MATERIAL DEFECTIVE	VENDOR
F08	SHIPPED EARLY/LATE	VENDOR
F09	PROBLEMS ADHESIVE/PSA	VENDOR
F10	FREIGHT INST. NO FOLLOWED	VENDOR
F11	SPECIAL INST. NO FOLLOWED	VENDOR
F12	OUT OF TOLERANCE	VENDOR
F13	PACKAGED INCORRECTLY	VENDOR
F14	DUROMETER DESC.	VENDOR
F15	COLOR DESC.	VENDOR
F16	WRONG THICKNESS	VENDOR
G01	A/R Adjustment	Corporate A/R
G02	\$5.00 or less balance	Corporate A/R
G03	Write off - 6 months	Corporate A/R
G04	Write off - Bankruptcy	Corporate A/R
G05	Write off - Collection Agency	Corporate A/R
G06	Refund	Corporate A/R
G07	Returned Check	Corporate A/R
H01	PROCESS CHANGE W/O CUST APPRVL	QUALITY CONTROL
H02	WRONG MATERIAL SHIPPED	QUALITY CONTROL
J01	PLOTTED INCORRECTLY	DIE SHOP / CAD DEP
J02	VERIFIED INCORRECTLY	DIE SHOP / CAD DEP
J03	TOOL BUILT INCORRECTLY	DIE SHOP / CAD DEP
M01	PACKING SLIPS SWITCHED	SHIP/PURCH/SCHED
M02	UPS LABELS SWITCHED	SHIP/PURCH/SCHED
M03	S/O NOT MOVED OUT-CANCELLED	SHIP/PURCH/SCHED
M04	DAMAGE DUE TO PACKAGING	SHIP/PURCH/SCHED
M05	MATL SHIPPED TO WRONG CUST.	SHIP/PURCH/SCHED
M06	WRONG FREIGHT CHARGES	SHIP/PURCH/SCHED

```
Alpha = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'V'];
Occ = [];
rjcodes = rj.read_csv("rejectcodes.csv")
for i = numel(Alpha)
    for t = 1:count(rjcodes)
        if rjcodes(t)[0] == Alpha(i)
            Occ(i) = Occ(i) + 1
        else
            Occ(i) = 0
        end
    end
end
end

NoFault = Occ(11)
Shipping = Occ(3) + Occ(10)
Production = Occ(2) + Occ(13)
Vendor = Occ(6) + Occ(17)
Engineering = Occ(14) + Occ(9)
Sales = Occ(1) + Occ(16);
Total = sum(Occ)

figure(1, figsize = (6,6));
ax = axes([0.1, 0.1, 0.8, 0.8]);
labels = 'No Fault', 'Shipping', 'Production', 'Vendor', 'Engineering', 'Sales';
fracs = [NoFault, Shipping, Production, Vendor, Engineering, Sales]/Total *100
explode = (0, 0.05, 0, 0)
pie(fracs, explode = explode, labels = labels,
autopct = '%1.1f%%', shadow = True, startangle = 90)
title ('Department Area NCR Contribution')
```

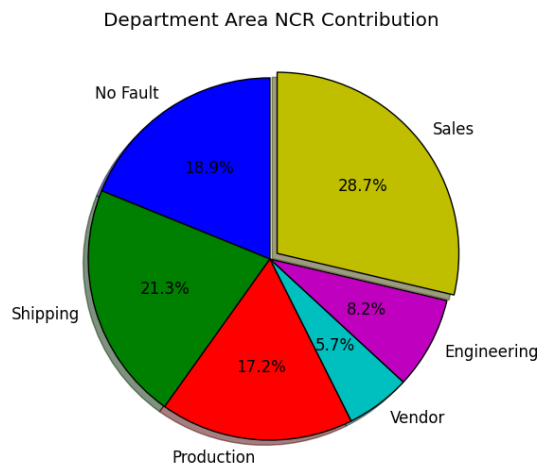


Figure: Output and Python Code

TEAM MEMBER CONTRIBUTIONS

- Carlos Oyuela-Mora:
 - EER diagram, Query 5 Design, Discussions with client, Business Overview
- Christopher Svenson:
 - Normalization, Access Implementation (Tables, Queries, Forms, Reports), EER, Presentation Design, Report Design, integration of content generated by members
- Qianwen Polly Zhang:
 - Helped with Normalization, EER diagram, relational design, scheduling meet times
- My Dinh:
 - EER Diagram, relational design, queries 3 and 4 design.
- Yian Liou:
 - Helped with Query justification, EER diagram, relational design, a little on normalization.
- John Shen:
 - Query 2 design, relational design, EER
- Kevin Khuu:
 - Helped with Normalization, Relational design of the relationship, EER diagram of customer
- Yitong Ellie Pan:
 - Query 1 design, relational design, EER
- Young Lin:
 - Helped with EER diagram, project schedule, relational design, normalization; wrote unused queries

DISCUSSION AND FUTURE WORK

Given the scale of the project, we were not able to complete a fully functioning database for Boyd Corporation. However, as our prototype does connect the basic workings of Boyd, our client is pleased and thrilled about the organizational and relatable skills the prototype demonstrates. As Boyd engages with the prototype, we can further improve the database to include more relevant information such as considering significant attributes within Corrective and Preventive Actions to affect the overall scorecard. Despite not having a fully operational database, the project is a positive and valuable first step. Our main client, the Director of Quality, Adriana Mora, must build up cases in order to validate certain directions she intends to take the company. Our database prototype is able to reinforce Mrs. Mora's ideas by simplifying the assembly and interpretation of data in Boyd's files allowing her to save time and collect more information.

Mrs. Mora followed up on her comments made after our final presentation and has asked us to work on implementing our project, queries and subsequent analysis in order to compare/benchmark and improve upon their new Salesforce database, possibly incorporating our design into that database on behalf of the Quality Assurance and Supply Chain departments. We are, of course, thrilled with this prospect and all, to the extent our academic obligations permit us, plan to take her up on this incredible offer.

This process has, for all of us, been one of (if not the) best team exercises we have embarked upon at Berkeley thus far. In particular, the team cohesion, individual engagement, individual effort, and commitment to this project could not have been better. While a good deal of this might just be luck of the draw in terms of getting good teammates, it is also at least in part due to how dynamic and challenging our client's needs were. We all picked a challenging project and were rewarded for it. It took a good deal of time and effort to ensure things did not get unwieldy or sidetracked and we feel rewarded simply having accomplished that. In general, for the continuation and actual implementation of this project as well as endeavors beyond it, we all hope to work together on other team efforts to come.