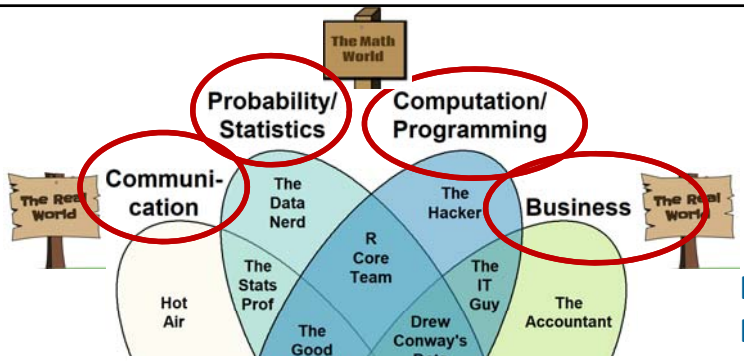
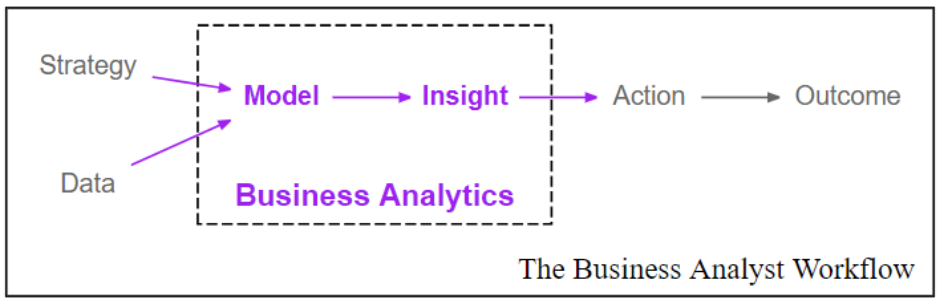
Graphical Models and Intro to greta/causact

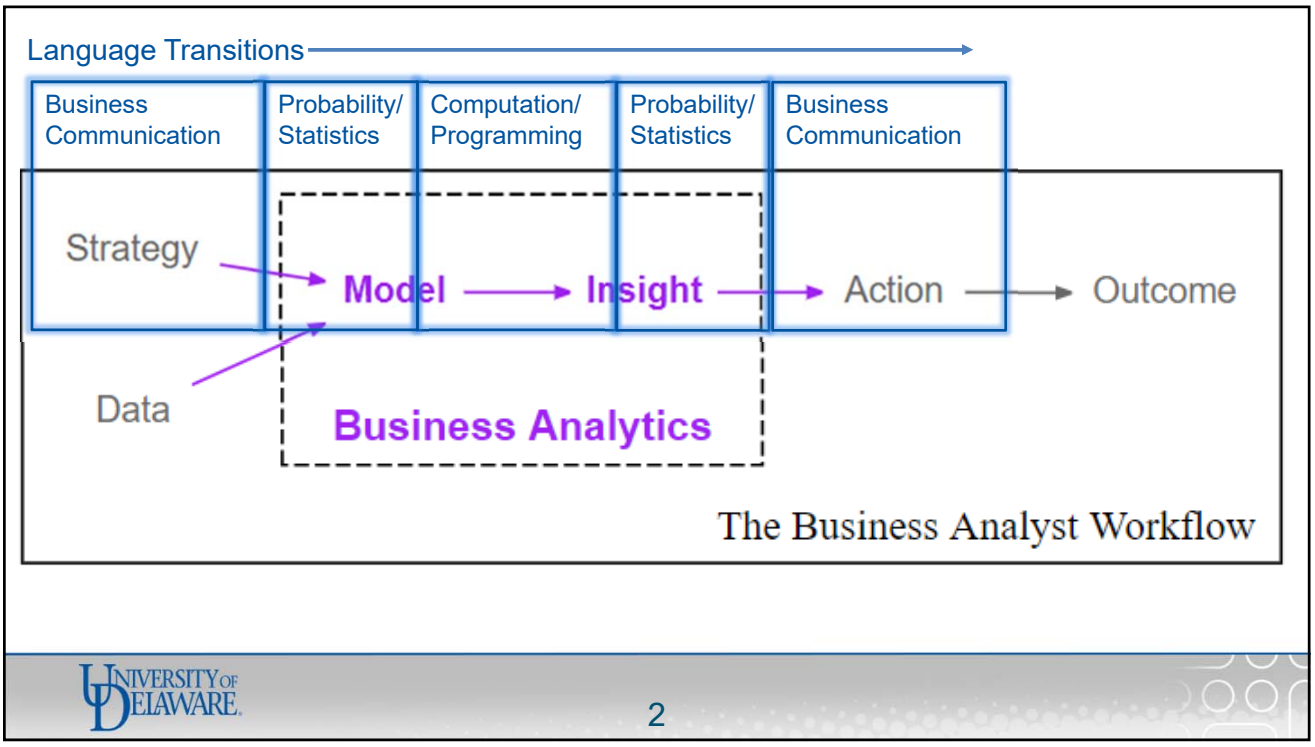
Speaking Data Analytics

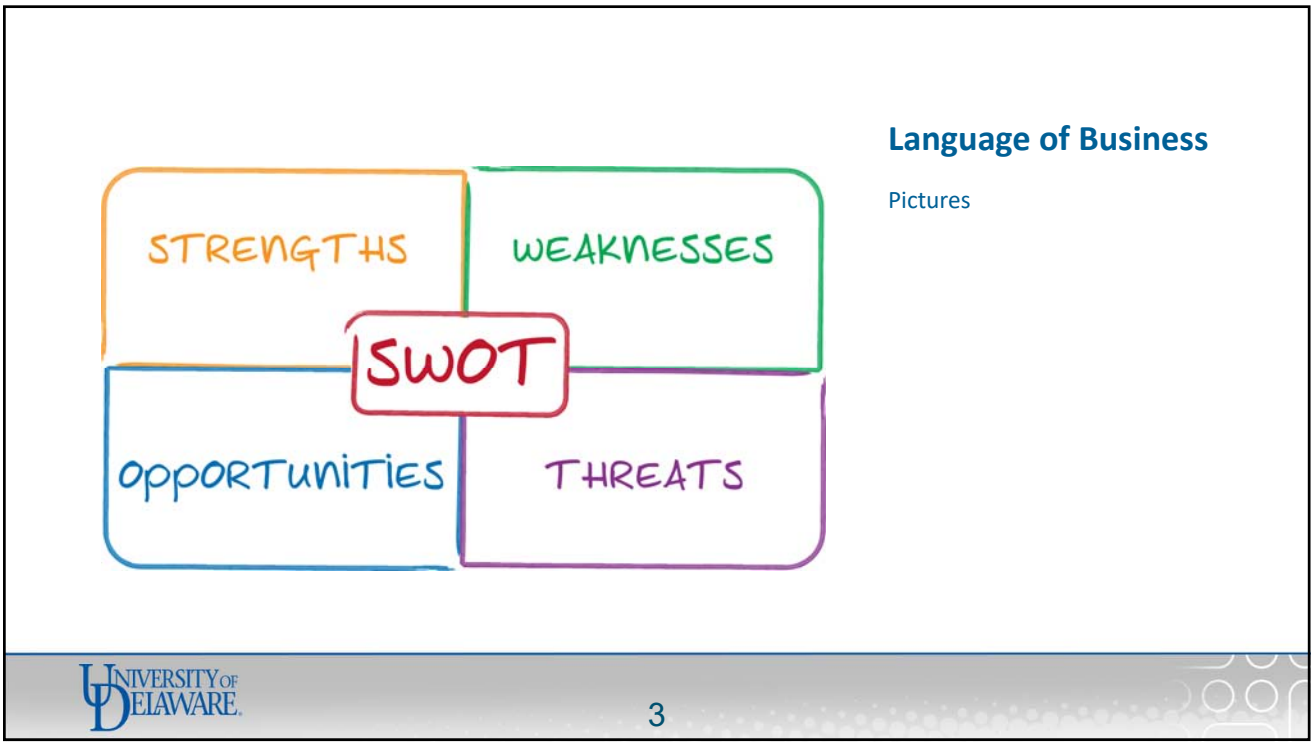


Data Science Venn Diagram



://bit.ly/2BjqdBt)


Language Transitions
Business Communication
Probability/ Statistics
Computation/ Programming
Probability/ Statistics
Business Communication
Strategy
Model
Insight
Action
Outcome
Data
Business Analytics
The Business Analyst Workflow
UNIVERSITY OF DELAWARE
2


STRENGTHS
WEAKNESSES
SWOT
OPPORTUNITIES
THREATS
Language of Business
Pictures
UNIVERSITY OF DELAWARE
3

$$
\begin{aligned}
K_i &\sim \text{Binomial}(n_i, p_i) \\
p_i &= \text{ilogit}(\alpha_j + \beta_j x_i) \\
\alpha_j &\sim \text{Normal}(\alpha, \sigma_\alpha) \\
\beta_j &\sim \text{Normal}(\beta, \sigma_\beta) \\
\alpha &\sim \text{Normal}(0, 10) \\
\sigma_\alpha &\sim \text{Uniform}(0, 5) \\
\beta &\sim \text{Normal}(0, 2) \\
\sigma_\beta &\sim \text{Uniform}(0, 5)
\end{aligned}
$$

Language of Statistics

Formulas

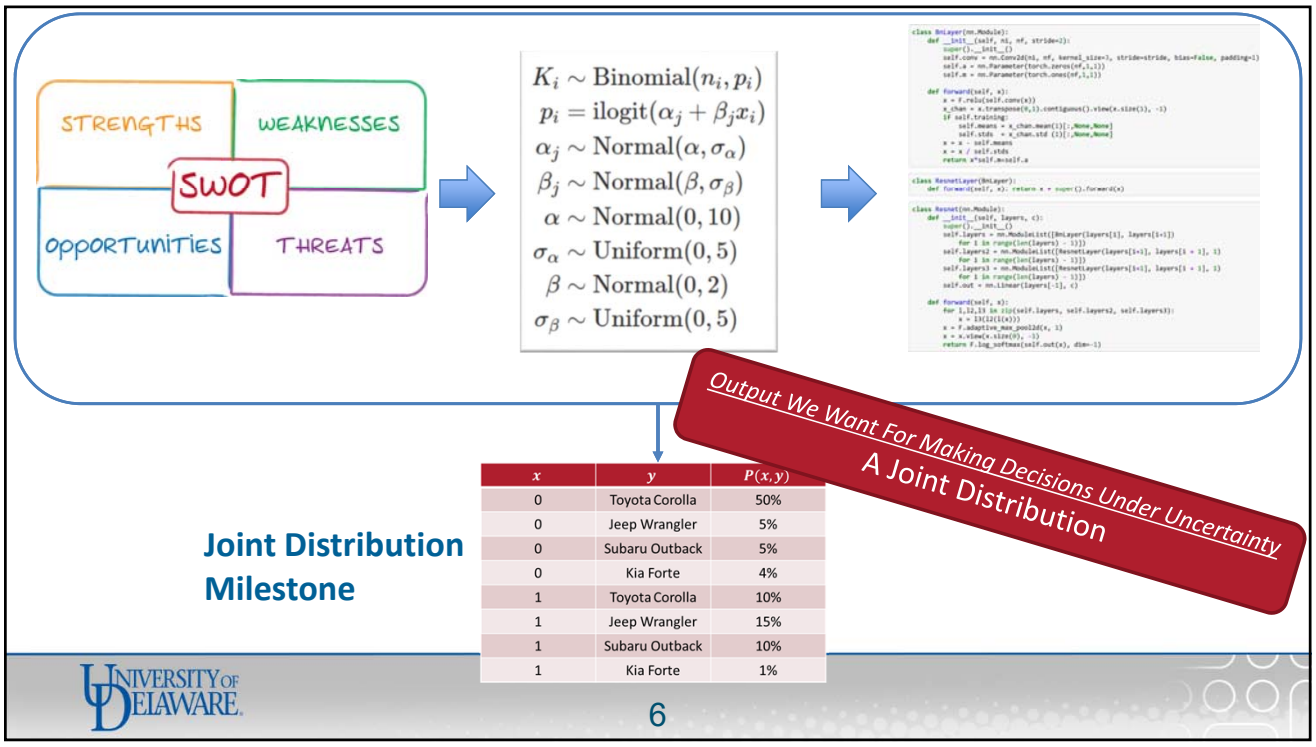

```python
class BnLayer(nn.Module):
    def __init__(self, ni, nf, stride=2):
        super().__init__()
        self.conv = nn.Conv2d(ni, nf, kernel_size=3, stride=stride, bias=False, padding=1)
        self.a = nn.Parameter(torch.zeros(nf,1,1))
        self.m = nn.Parameter(torch.ones(nf,1,1))

    def forward(self, x):
        x = F.relu(self.conv(x))
        x_chan = x.transpose(0,1).contiguous().view(x.size(1), -1)
        if self.training:
            self.means = x_chan.mean(1)[:,None,None]
            self.stds  = x_chan.std (1)[:,None,None]
        x = x - self.means
        x = x / self.stds
        return x*self.m+self.a
```

```python
class ResnetLayer(BnLayer):
    def forward(self, x): return x + super().forward(x)
```

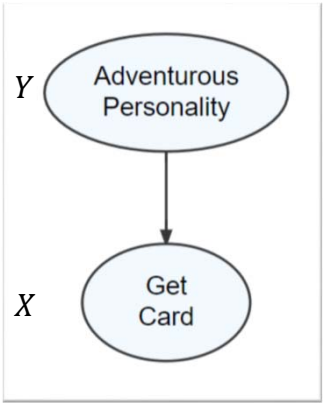
```python
class Resnet(nn.Module):
    def __init__(self, layers, c):
        super().__init__()
        self.layers = nn.ModuleList([BnLayer(layers[i], layers[i+1])
            for i in range(len(layers) - 1)])
        self.layers2 = nn.ModuleList([ResnetLayer(layers[i+1], layers[i + 1], 1)
            for i in range(len(layers) - 1)])
        self.layers3 = nn.ModuleList([ResnetLayer(layers[i+1], layers[i + 1], 1)
            for i in range(len(layers) - 1)])
        self.out = nn.Linear(layers[-1], c)

    def forward(self, x):
        for l,l2,l3 in zip(self.layers, self.layers2, self.layers3):
            x = l3(l2(l(x)))
        x = F.adaptive_max_pool2d(x, 1)
        x = x.view(x.size(0), -1)
        return F.log_softmax(self.out(x), dim=-1)
```

Language of Computers

Code



x	y	$P(x, y)$
0	Toyota Corolla	50%
0	Jeep Wrangler	5%
0	Subaru Outback	5%
0	Kia Forte	4%
1	Toyota Corolla	10%
1	Jeep Wrangler	15%
1	Subaru Outback	10%
1	Kia Forte	1%



$$P(Y)$$

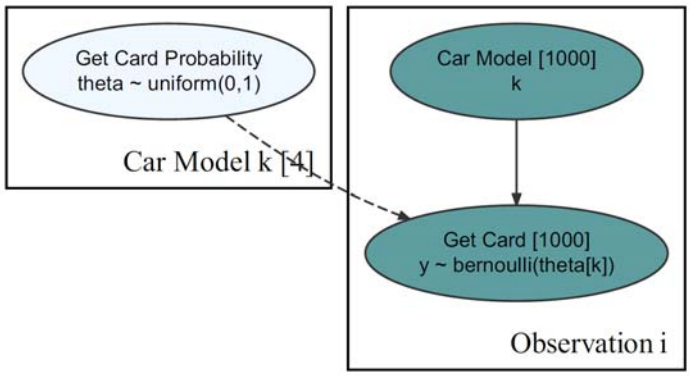
$$P(X|Y)$$

$$P(X,Y) = P(Y) \times P(X|Y)$$

The DAG

Nodes are random variables

Edges drawn from parent to child indicate a node's uncertainty is expressed conditionally on its parents



CAUSACT and GRETA Demo

Plates represent a duplication of the random variables inside of them. One per member of the plate's index. For example, there are four car models, hence we will get four theta values.

Mathematical models of distributional assumptions of the random variable are shown.

Walk through R-Code files:

`DAGModellingWalkthrough.R`
`betaDist.R`
`creditCard.R`