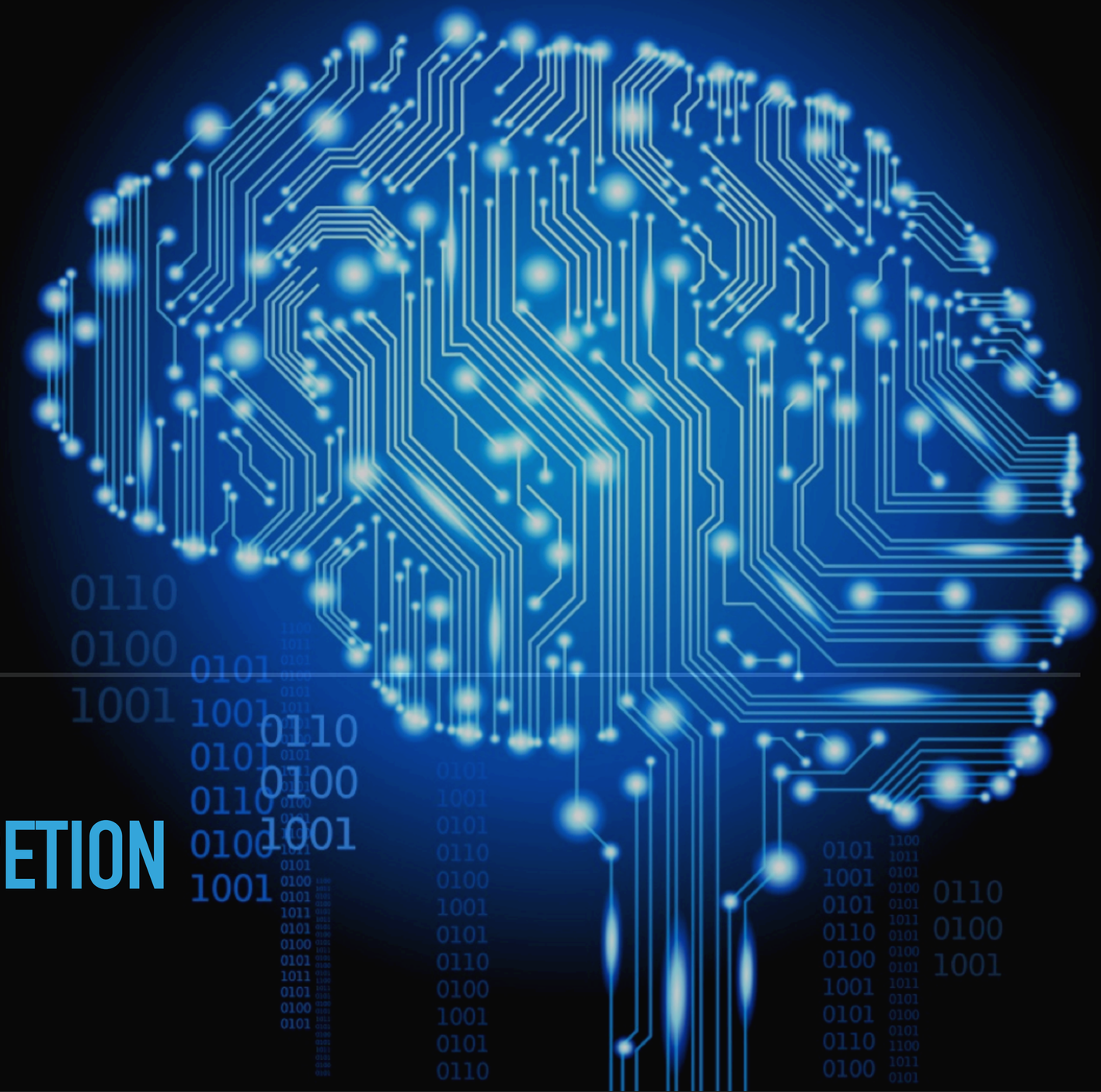
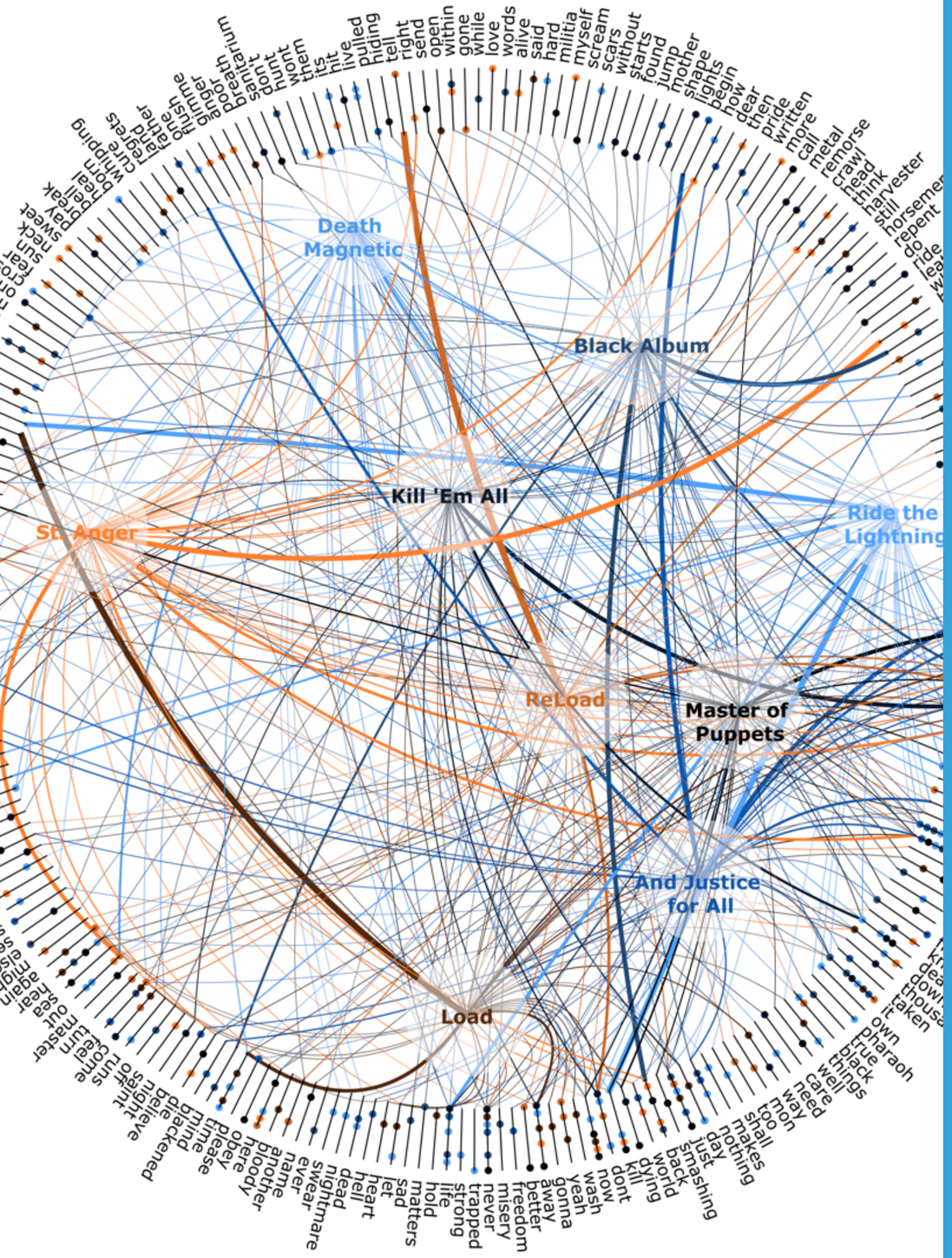
FEATURE ENGINEERING

HOW TO WIN A KAGGLE COMPETION



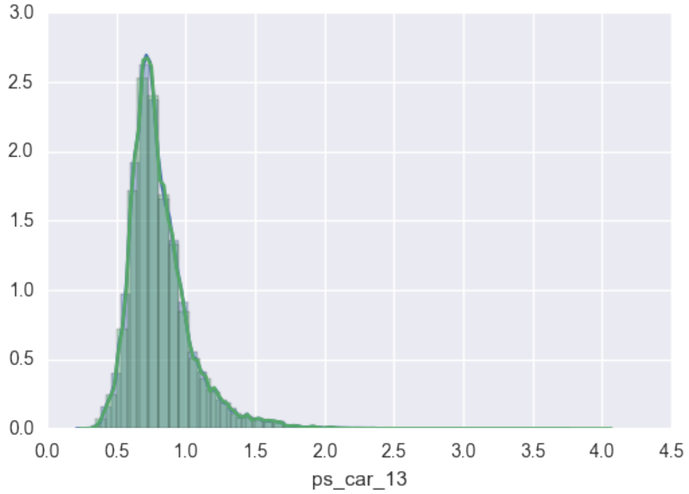


EDA (EXPLORATORY DATA ANALYSIS)

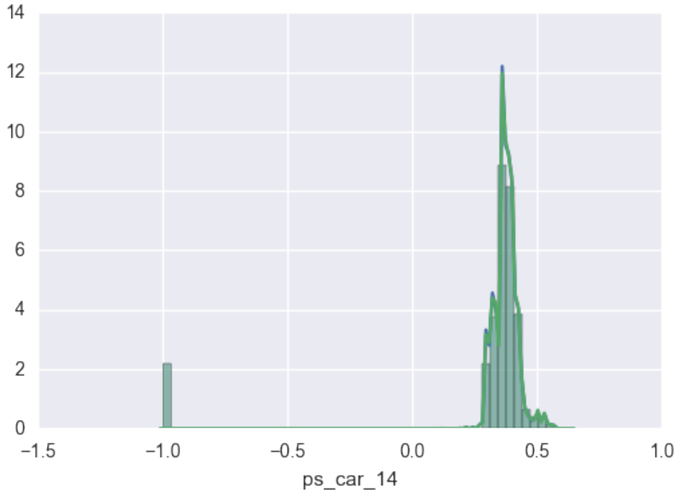
- Maximize insights
- Uncover hidden patterns
- Extract important features
- Detect outliers and anomalies
- Validate underlying assumptions

FEATURE DISTRIBUTION

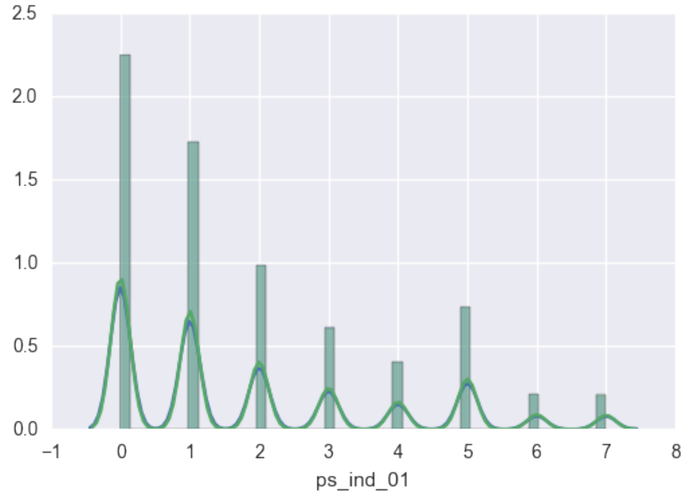
- Continuous feature
- Normally distributed with outliers



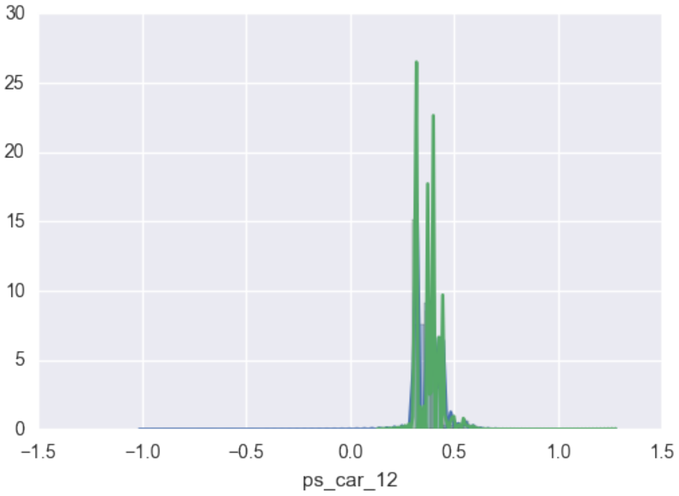
- Continuous feature
- Normally distributed with missing values imputed as -1



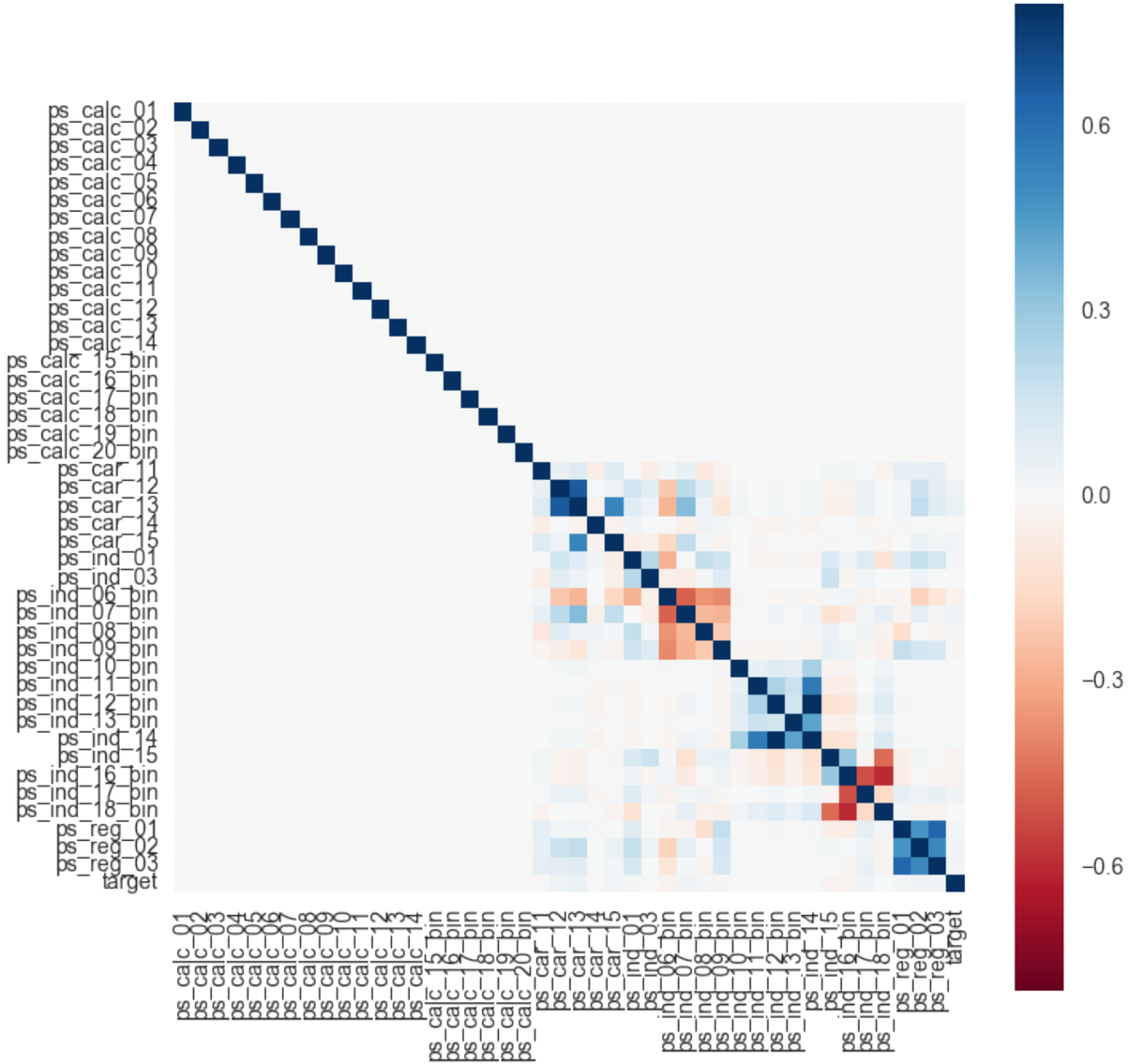
- Discrete feature



- Continuous feature



FEATURE CORRELATION




```
corrmat = train_data[num_vars+[target_var]].dropna().corr()
f, ax = plt.subplots(figsize=(8, 8))
sns.heatmap(corrmat, vmax=.8, square=True, cmap='RdBu');
```

- Registrant related features are highly correlated to target
- Calculated features seem to have little to zero correlation to target. Maybe we can remove them from the datasets?

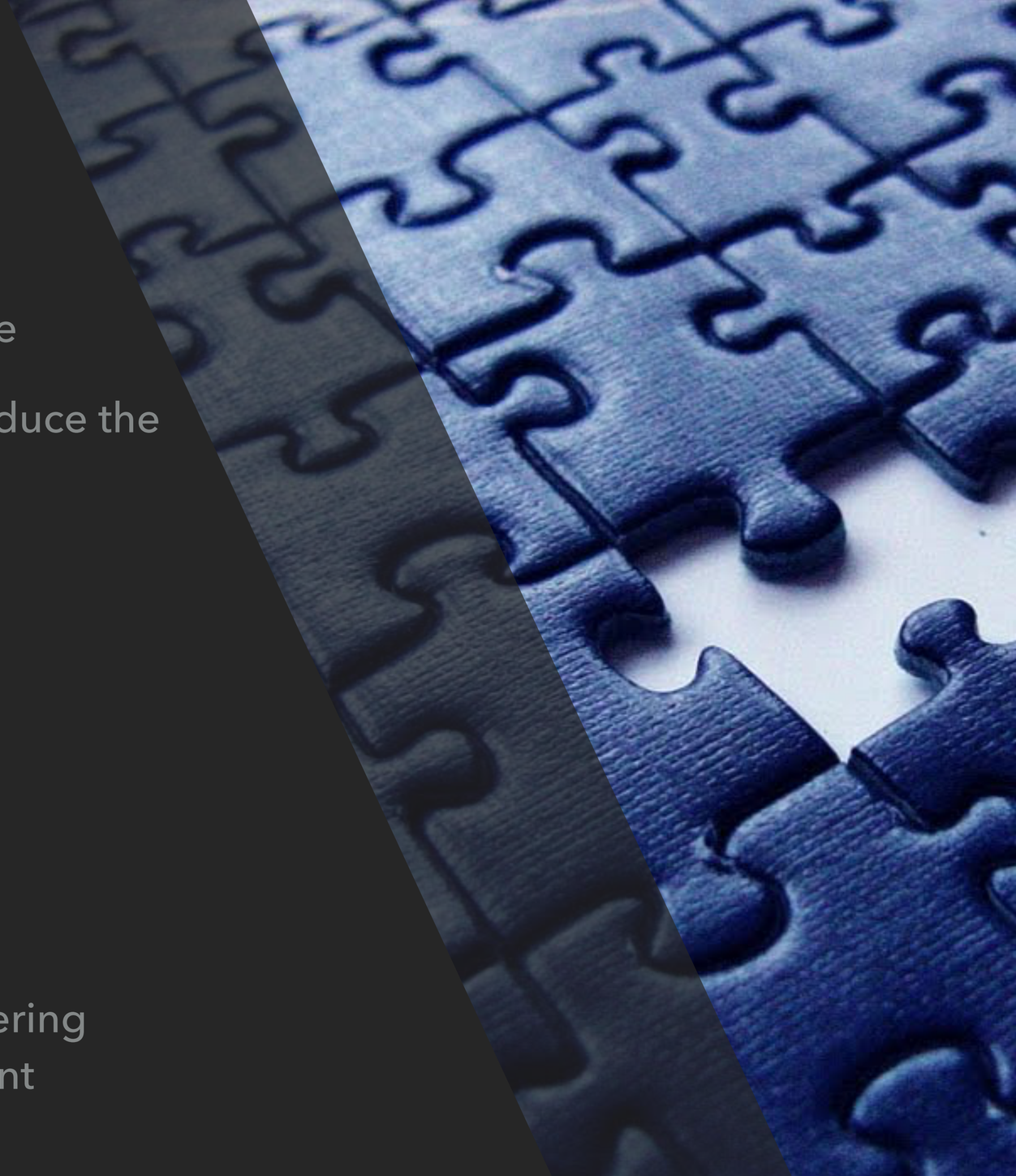
FEATURE ENGINEERING

COMING UP WITH FEATURES IS DIFFICULT, TIME-CONSUMING, REQUIRES EXPERT KNOWLEDGE. "APPLIED MACHINE LEARNING" IS BASICALLY FEATURE ENGINEERING.

Andrew Ng

MISSING VALUES IMPUTATION

- **Numeric Features**
  - Impute with mean for normal distributed feature
  - Impute with median for skewed distribution (reduce the impacts of outliers)
- **Categorical Features**
  - Majority imputation
- **Supervised learning imputation**
  - Regression/ Classification
- **XGBoost/LightGBM:**
  - Missing values are gracefully treated by considering missing value as candidate for tree splitting point



DATA PREPROCESSING – NUMERIC FEATURES

- Standardization
  - To standardize features by removing the mean and scaling to unit variance
  - sklearn.preprocessing.StandardScaler
  - Required by SVM/ K-means. Helps linear models such as Linear Regression, Logistic Regression, LASSO/Ridge and NN to converge faster.
  - Not needed by tree models

$$x_{new} = \frac{x - \mu}{\sigma}$$

```
from sklearn.preprocessing import StandardScaler

data = [[0, 0], [0, 0], [1, 1], [1, 1]]
scaler = StandardScaler()
print(scaler.fit(data))
```

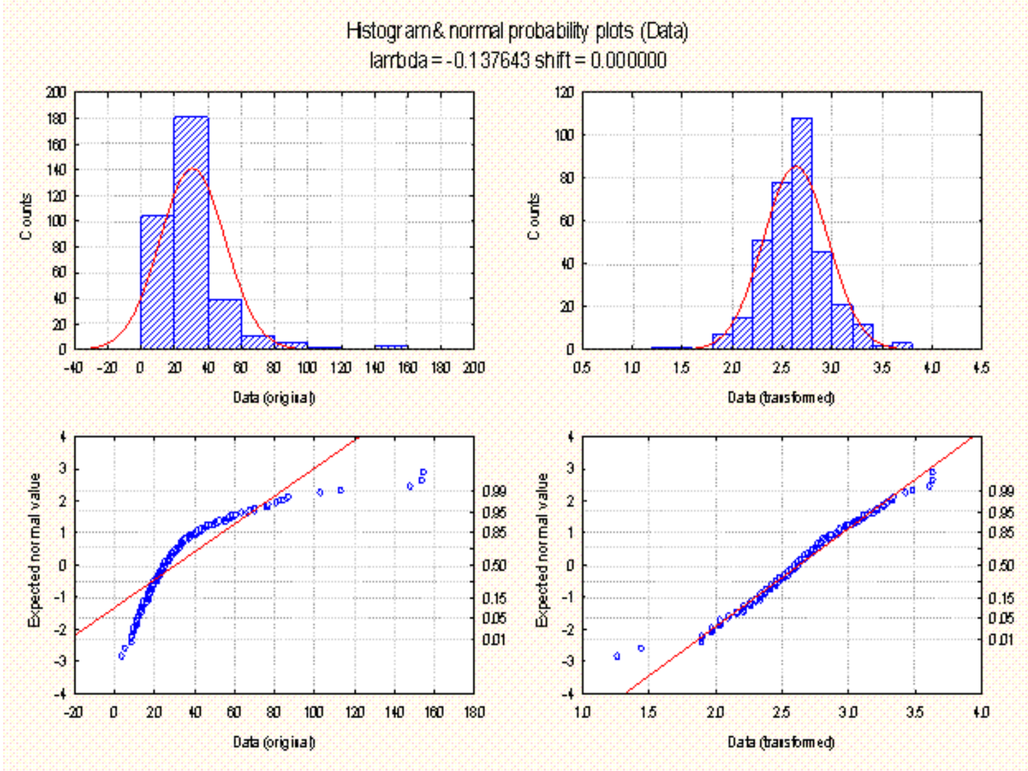
DATA PREPROCESSING – NUMERIC FEATURES

- **Normalization**
  - Scale individual samples to have unit norm (e.g. 0-1)
  - sklearn.preprocessing.Normalizer
  - Helps models such as Linear Regression, Logistic Regression, LASSO/Ridge and NN to converge faster.
  - Not needed by tree models

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

DATA PREPROCESSING – NUMERIC FEATURES

- Box-Cox transformation
  - Reduce data skewness by shifting the distribution
  - $$y = \begin{cases} {(x^{\lambda}) - 1}/{\lambda} & when\ \lambda \neq 0 \\ \log(x) & when\ \lambda = 0 \end{cases}$$
  - scipy.stats.boxcox
  - Requires the input data to be positive
  - Not needed by tree models



DATA PREPROCESSING – CATEGORICAL FEATURES

- **One Hot Encoding**
- Encode categorical **integer** features using a one-hot aka one-of-K schemeLabel Encoding
- The most adaptive encoding approach. Works for both linear models and tree models
- [sklearn.preprocessing.OneHotEncoder]
- Categorical text features need to be transformed to integers prior to OHE.
- Sparse matrix is the recommended format for outputs

Sample	Category	Numerical
1	Human	1
2	Human	1
3	Penguin	2
4	Octopus	3
5	Alien	4
6	Octopus	3
7	Alien	4

OHE

Sample	Human	Penguin	Octopus	Alien
1	1	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	0	0	1	0
7	0	0	0	1

```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()
enc.fit([[0, 0, 3], [1, 1, 0], [0, 2, 1], [1, 0, 2]])
```

DATA PREPROCESSING – CATEGORICAL FEATURES

- **Label Encoding**
- Encode categorical features with integers between 0 and # of levels -1
- Works for tree models such as Random Forest, GBDT, GBM, XGBoost and LightGBM
- Not work for linear models
- sklearn.preprocessing.OneHotEncoder
- Pros:
  - RAM efficient
  - Works for categorical features with large number of levels
  - Easier to implement

Sample	Category	Numerical
1	Human	1
2	Human	1
3	Penguin	2
4	Octopus	3
5	Alien	4
6	Octopus	3
7	Alien	4

Label-encoded

DATA PREPROCESSING – CATEGORICAL FEATURES

- **Mean encoding**
- Encode high-cardinality categorical features by processing train targets (y) with Empirical Bayes.
- Most effective encoding approach for high-cardinality categorical features
- Works for both linear and tree models
- It's a supervised learning approach and is prone to overfitting thus needs to be carefully tuned
- A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems
- 平均数编码：针对高基数定性特征（类别特征）的数据预处理/特征工程

Prior prob: $P(y = target)$

Posterior prob: $P(target = y|variable = k)$

For any given sample *k*, the estimated prob is:

$$\hat{P} = \lambda * \hat{P}(y = target) + (1 - \lambda) * \hat{P}(target = y|variable = k)$$

Where lambda is defined as :

$$\lambda(n) = \frac{1}{1 + e^{(n-k)/f}}$$

k and f are the parameters to be tuned.

FEATURE IMPORTANCE ANALYSIS/ FEATURE SELECTION

- **Low-variance filter**
  - Works for both tree and linear models
- **High-correlation filter**
  - Works for linear model
  - May lead to feature engineering ideas for tree models
- **Recursive Feature Elimination**
  - Forward RFE: **add** one feature at a time until validation score doesn't improve
  - Backward RFE: **remove** one feature a time until validation score doesn't improve
- **Coefficient/ Importance**
  - Linear Regression
  - Random Forest
  - XGBoost/LightGBM/GBM/GBDT


QUESTIONS?