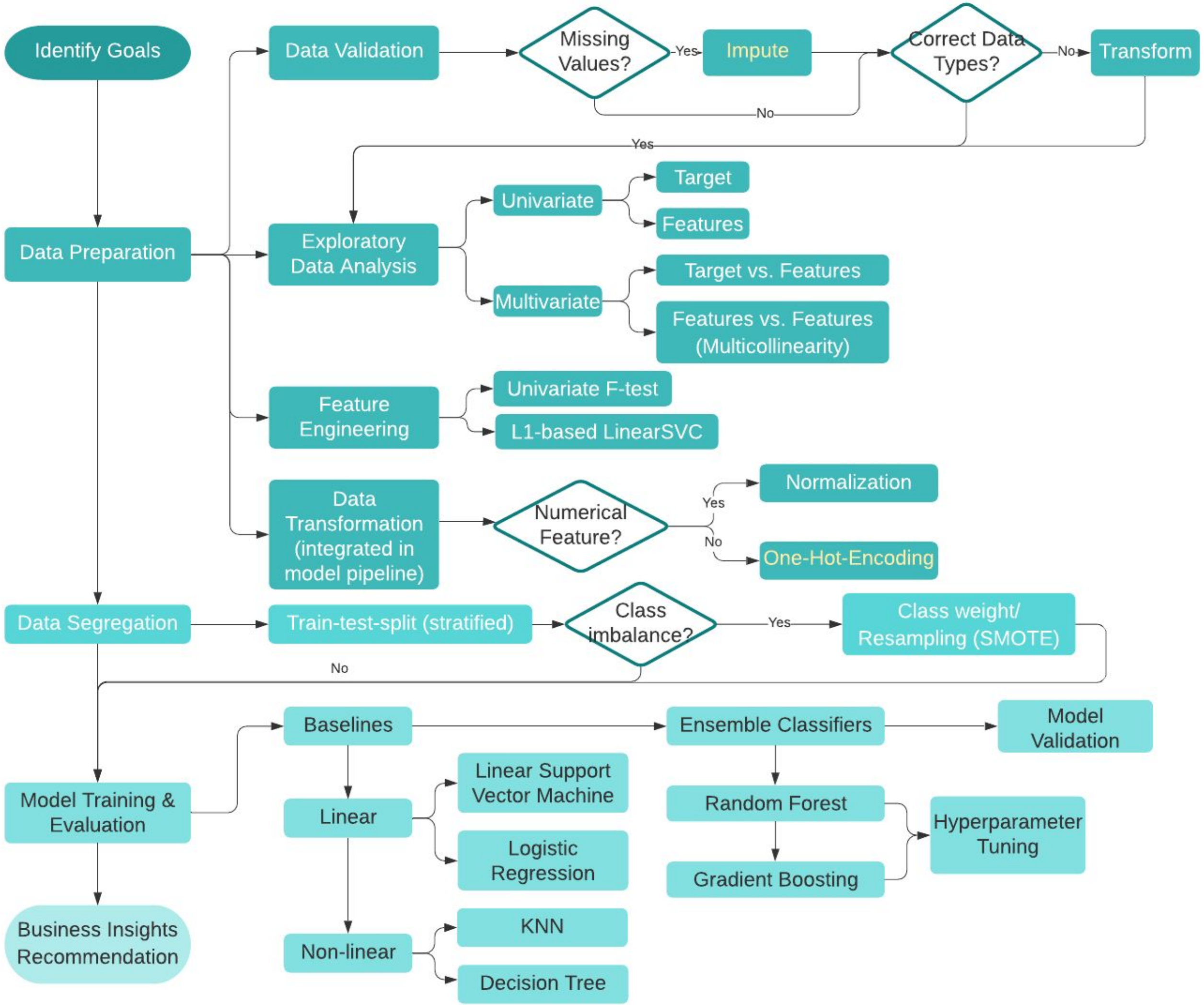
Search Engine Relevance Optimization

Qi Feng

Overview


Identify Goals
Data Validation
Missing Values?
Yes
Impute
No
Correct Data Types?
No
Transform
Yes
Data Preparation
Exploratory Data Analysis
Univariate
Target
Features
Multivariate
Target vs. Features
Features vs. Features (Multicollinearity)
Feature Engineering
Univariate F-test
L1-based LinearSVC
Data Transformation (integrated in model pipeline)
Numerical Feature?
Yes
Normalization
No
One-Hot-Encoding
Data Segregation
Train-test-split (stratified)
Class imbalance?
Yes
Class weight/ Resampling (SMOTE)
No
Model Training & Evaluation
Baselines
Linear
Linear Support Vector Machine
Logistic Regression
Non-linear
KNN
Decision Tree
Ensemble Classifiers
Random Forest
Gradient Boosting
Hyperparameter Tuning
Model Validation
Business Insights Recommendation

Identify Goals

- Provide a solution (a classification model) for optimizing search engine relevance
- We want our model to be able to identify relevant responses
  - And minimize irrelevant responses that are predicted relevant

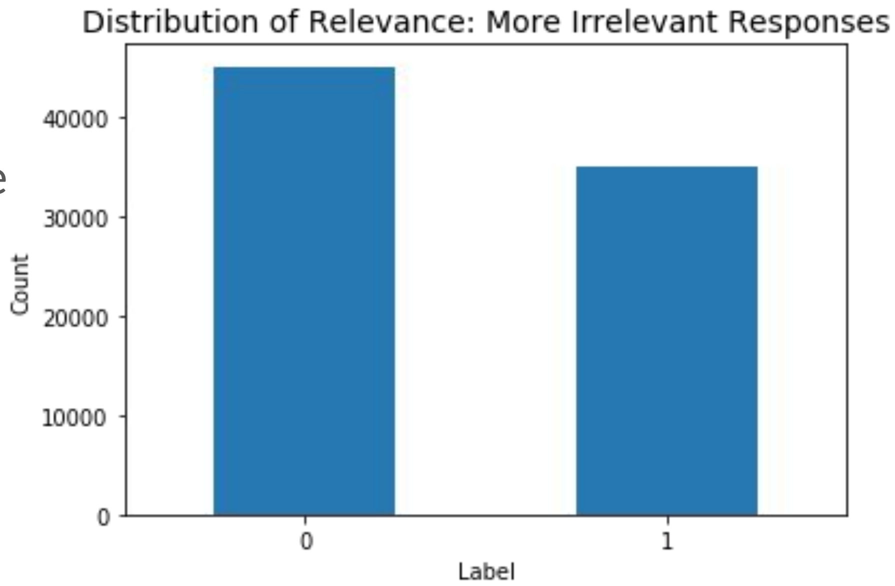
Data

- 80046 observations (responses)
- 12 features, 1 target
- No missing value

	query_id	url_id	query_length	is_homepage	sig1	sig2	sig3	sig4	sig5	sig6	sig7	sig8	relevance
0	4631	28624	2	1	0.09	0.15	1288	352	376	13	0.46	0.35	0
1	4631	28625	2	1	0.20	0.35	4662	337	666	28	0.43	0.27	1
2	4631	28626	2	1	0.36	0.49	1121	385	270	15	0.34	0.20	1
3	4631	28627	2	1	0.21	0.45	2925	478	640	14	0.44	0.33	1
4	4631	28628	2	1	0.25	0.42	1328	429	412	27	0.40	0.57	1

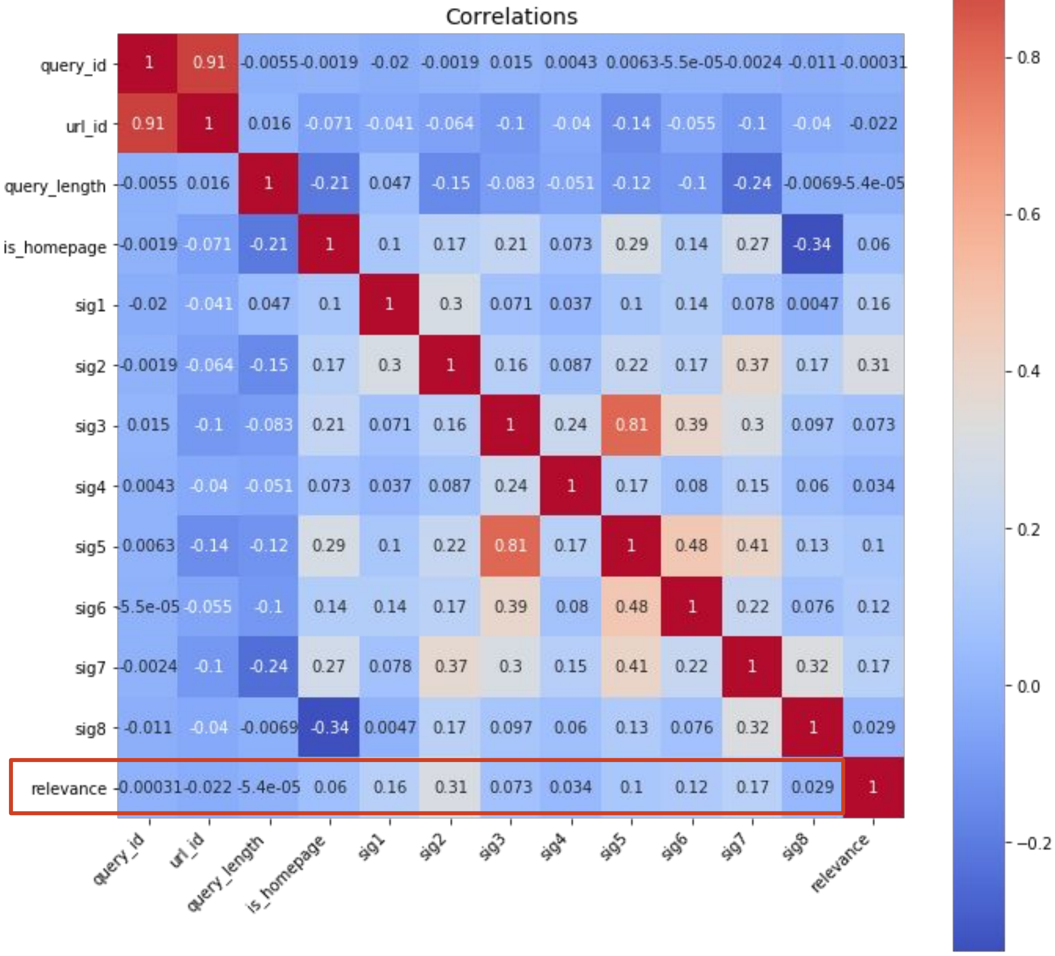
Target

- **Target:** `relevance`. Whether the search result is relevant or not.
    - 0 = Irrelevant
    - 1 = Relevant



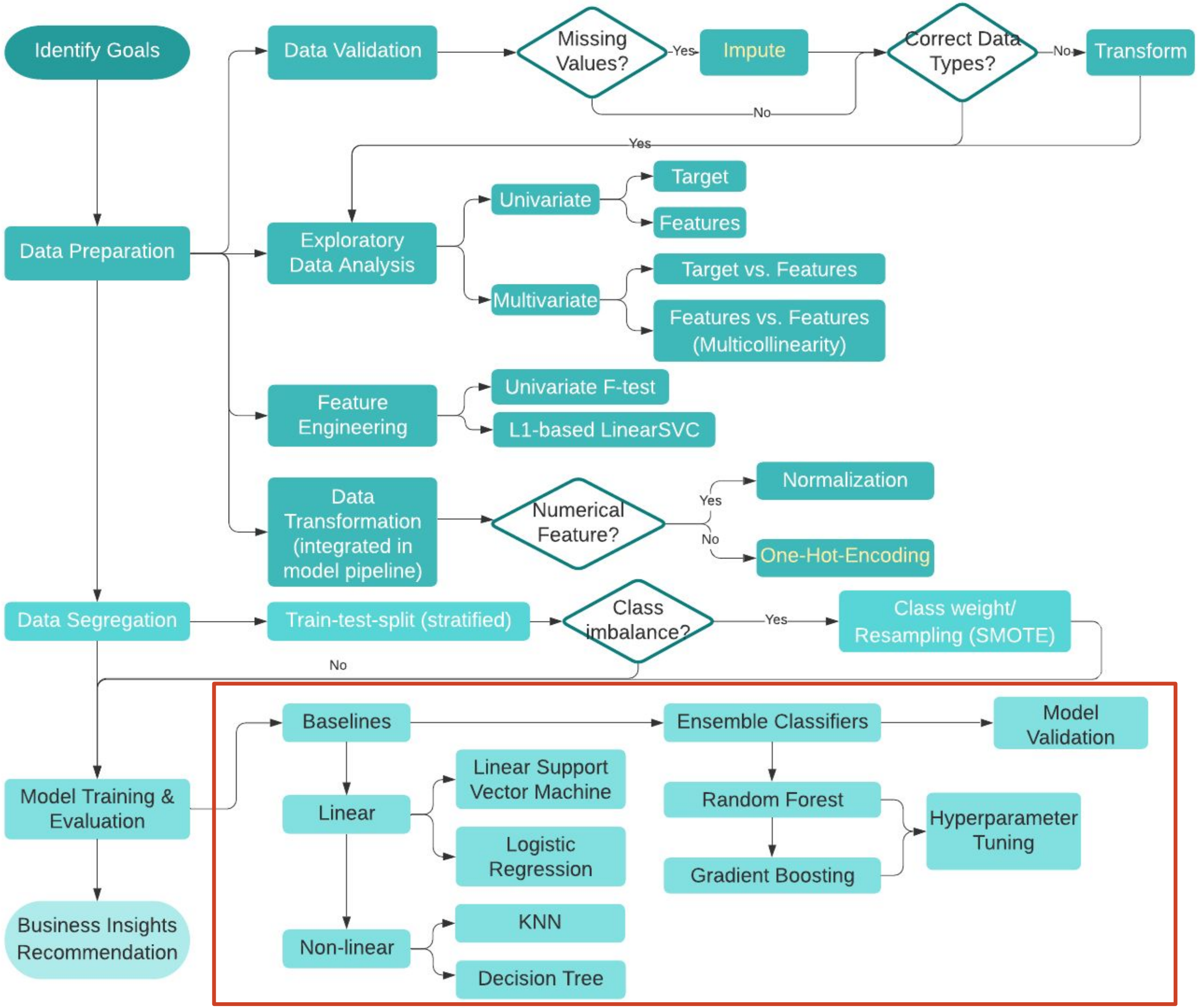
Features

- 3 Categorical
  - `query_id, url_id`: dropped
  - `is_homepage`
- 4 Continuous: `sig1, sig2, sig7, sig8`
  - Range in (0, 1)
- 5 Discrete:
  - `query_length`: most queries are short (of length 1-5)
  - `sig3, sig4, sig5, sig6`
    - Varying scales
    - Outliers



- Weak correlations of `relevance` vs. `query_length/query_id`
- Strongest positive correlation of `Relevance` vs. `sig2`
- Multicollinearity

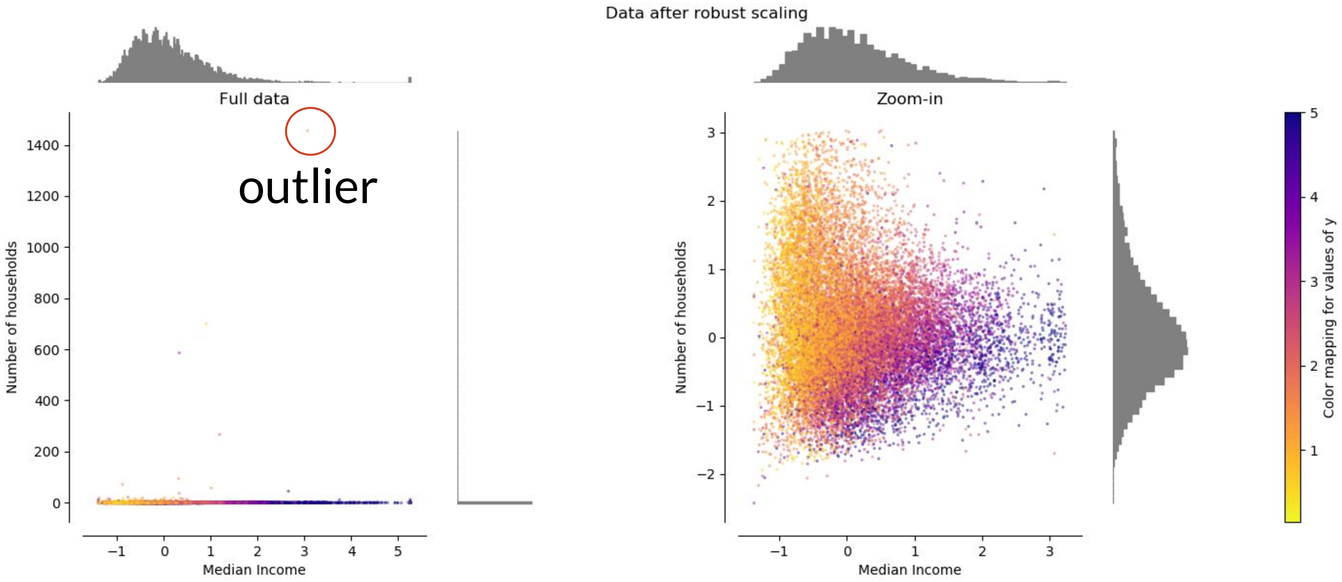
Modeling


Identify Goals
Data Validation
Missing Values?
Yes
Impute
No
Correct Data Types?
No
Transform
Yes
Data Preparation
Exploratory Data Analysis
Univariate
Target
Features
Multivariate
Target vs. Features
Features vs. Features (Multicollinearity)
Feature Engineering
Univariate F-test
L1-based LinearSVC
Data Transformation (integrated in model pipeline)
Numerical Feature?
Yes
Normalization
No
One-Hot-Encoding
Data Segregation
Train-test-split (stratified)
Class imbalance?
Yes
Class weight/ Resampling (SMOTE)
No
Model Training & Evaluation
Baselines
Linear
Linear Support Vector Machine
Logistic Regression
Non-linear
KNN
Decision Tree
Ensemble Classifiers
Random Forest
Gradient Boosting
Hyperparameter Tuning
Model Validation
Business Insights Recommendation

Preprocessing in Model Pipeline

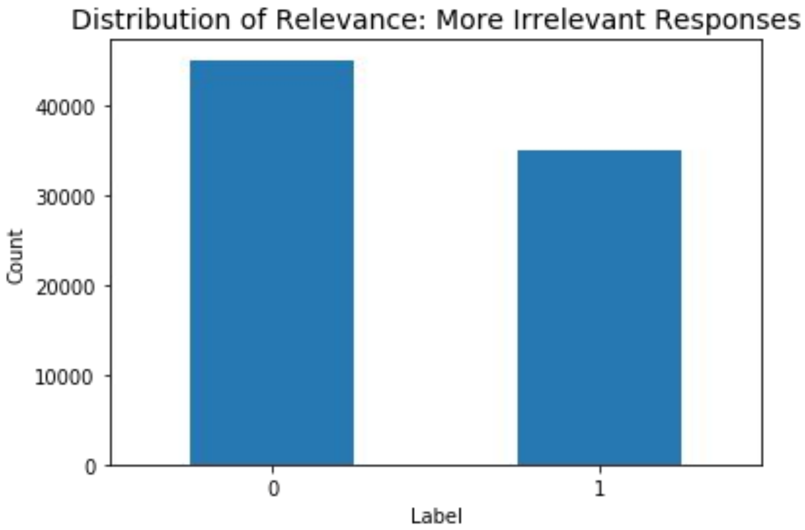
- Scaling numerical features
  - RobustScaler
- Dealing with class imbalance
  - Class weight
- Preprocessing boosts validation ROC-AUC by 24.5%

Scaling: RobustScaler



Class Imbalance

- Problem: unsatisfactory classifiers
- Solution
  - Data: Class weight
  - Evaluation metric
    - [ROC AUC]
  - Model
    - Bagging/Boosting-based models, e.g., Random Forest/Gradient Boosting.



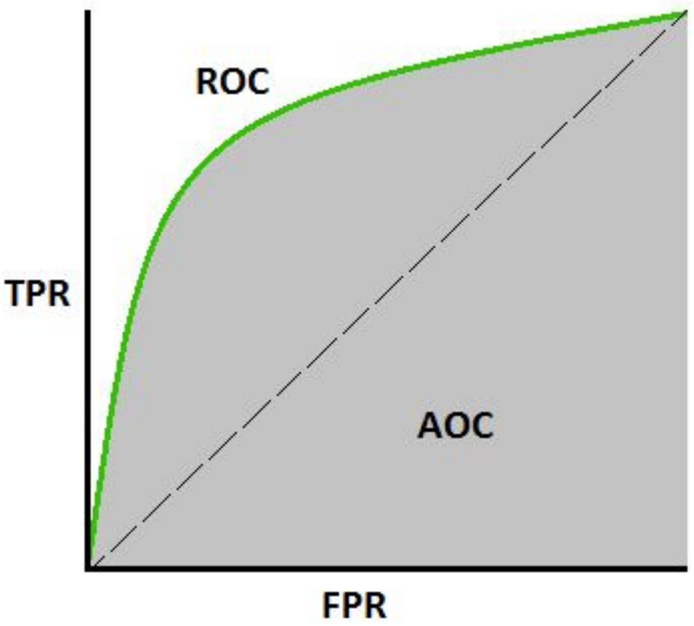
ROC-AUC

- Performance measurement for classification problem
- ROC is a probability curve and AUC is the area under the ROC curve
- How much the model is capable of distinguishing between classes

predicted→ real↓	*Class_pos*	*Class_neg*
Class_pos	TP	FN
Class_neg	FP	TN

$$\text{TPR (sensitivity)} = \frac{TP}{TP+FN}$$

$$\text{FPR (1-specificity)} = \frac{FP}{TN+FP}$$

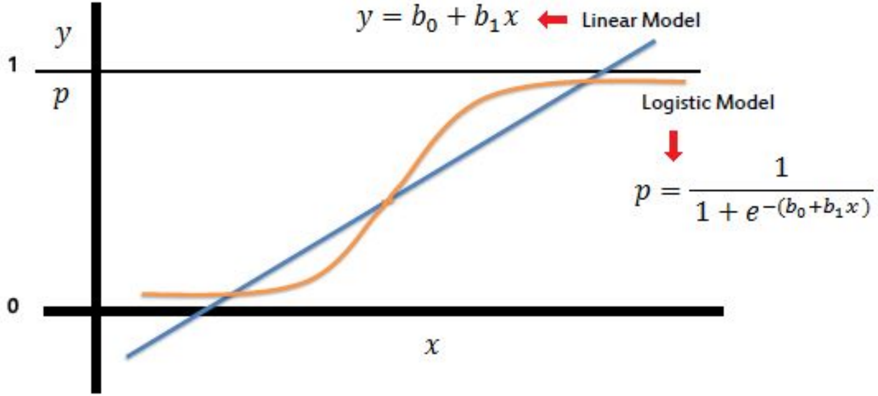


Baselines

For benchmarking performances

- Linear
  - Logistic Regression
  - Linear SVM
- Non-linear
  - KNN
  - Decision Tree

Best Baseline: Logistic Regression



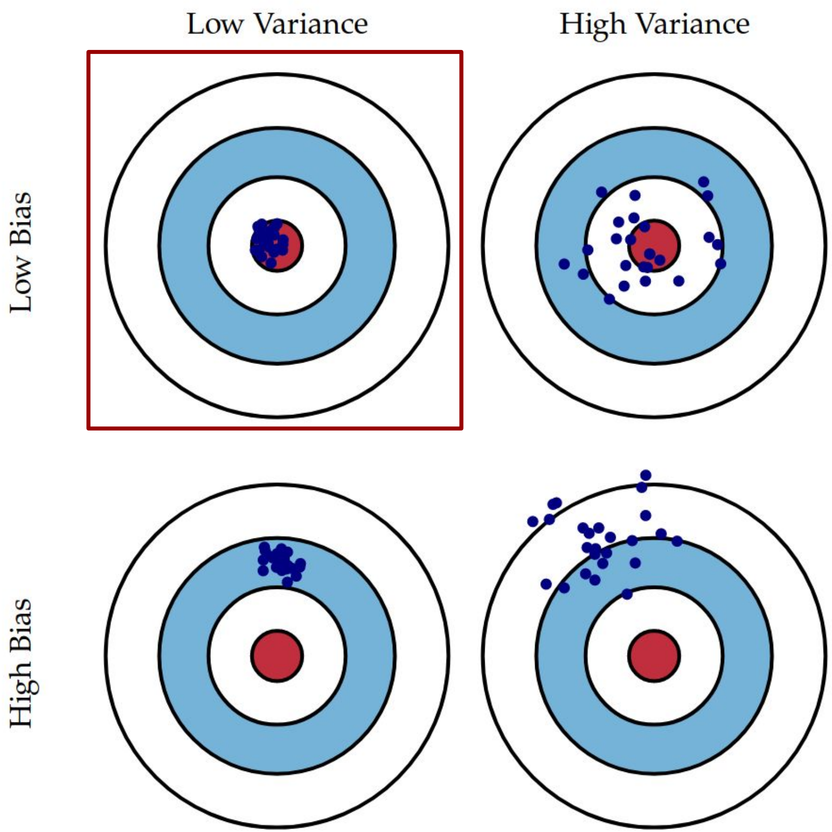
- Models linear relationship between the logit of the outcome probability and the independent variable
- **Pros**
  - **Probabilistic interpretation**
  - **Allows regularization**
  - **Simple**
- Cons
  - Does not fit well on large # of feature space or categorical features
  - Need transformations of non-linear features

Logit: $ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 x$

=> $P = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}$

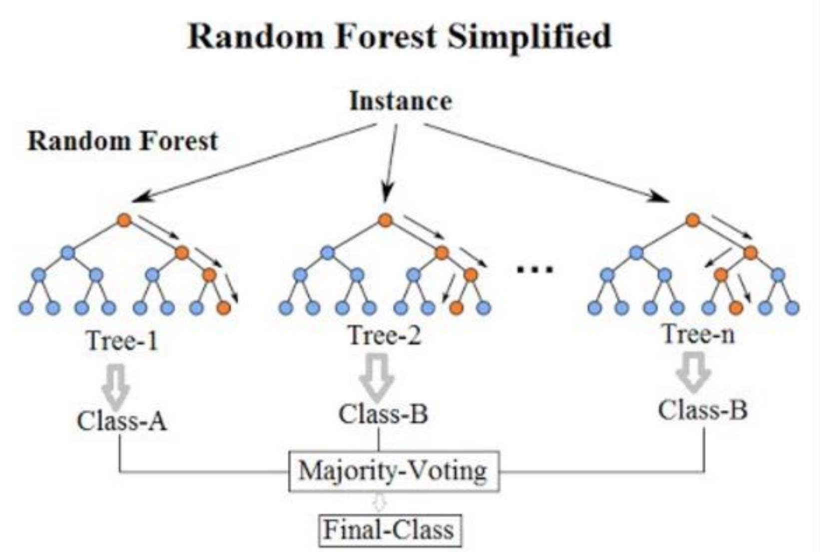
Ensembles

- Two families
    - Averaging
        - Random Forest
    - Boosting
        - XGBoost



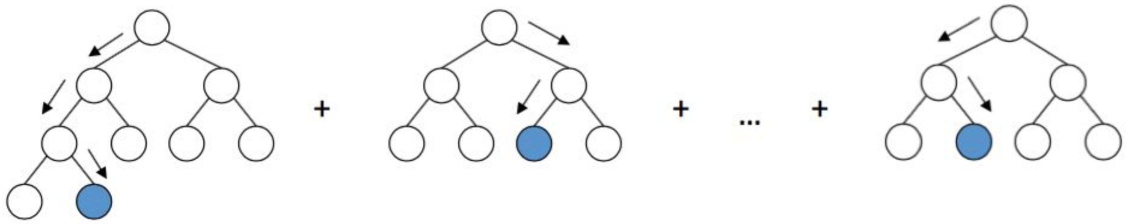
Random Forest

- Pros
    - **Prevents overfitting (reduces variance in a single decision tree)**
    - **No need of preprocessing**
    - **Flexible, usually high accuracy**
- Cons
    - Extrapolation effect
    - Complexity
    - Slow training




```
Validation Scores:
        metric  val_score
3      roc_auc   0.715073
2       recall   0.606096
1    precision   0.612988
0           f1   0.609490
Execution Time: 361.26 s
```

XGBoost



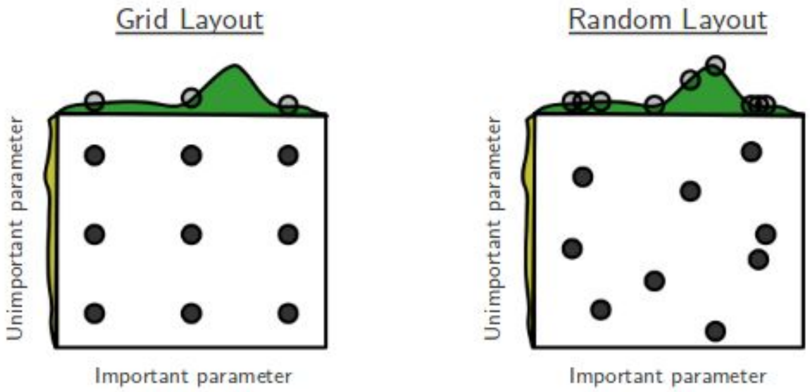
Validation Scores:

	metric	val_score
3	roc_auc	0.716682
2	recall	0.616064
1	precision	0.611434
0	f1	0.613682

Execution Time: 125.19 s

- Gradient Boosting: minimize loss by adding weak learners iteratively
- XGBoost: gradient boosting with more accurate approximations to find the best tree model
- **Pros**
  - **Fast training and prediction**
  - **Highly flexible, typically more accurate than Random Forest**
- Cons
  - Needs hyperparameter tuning
  - Less interpretable

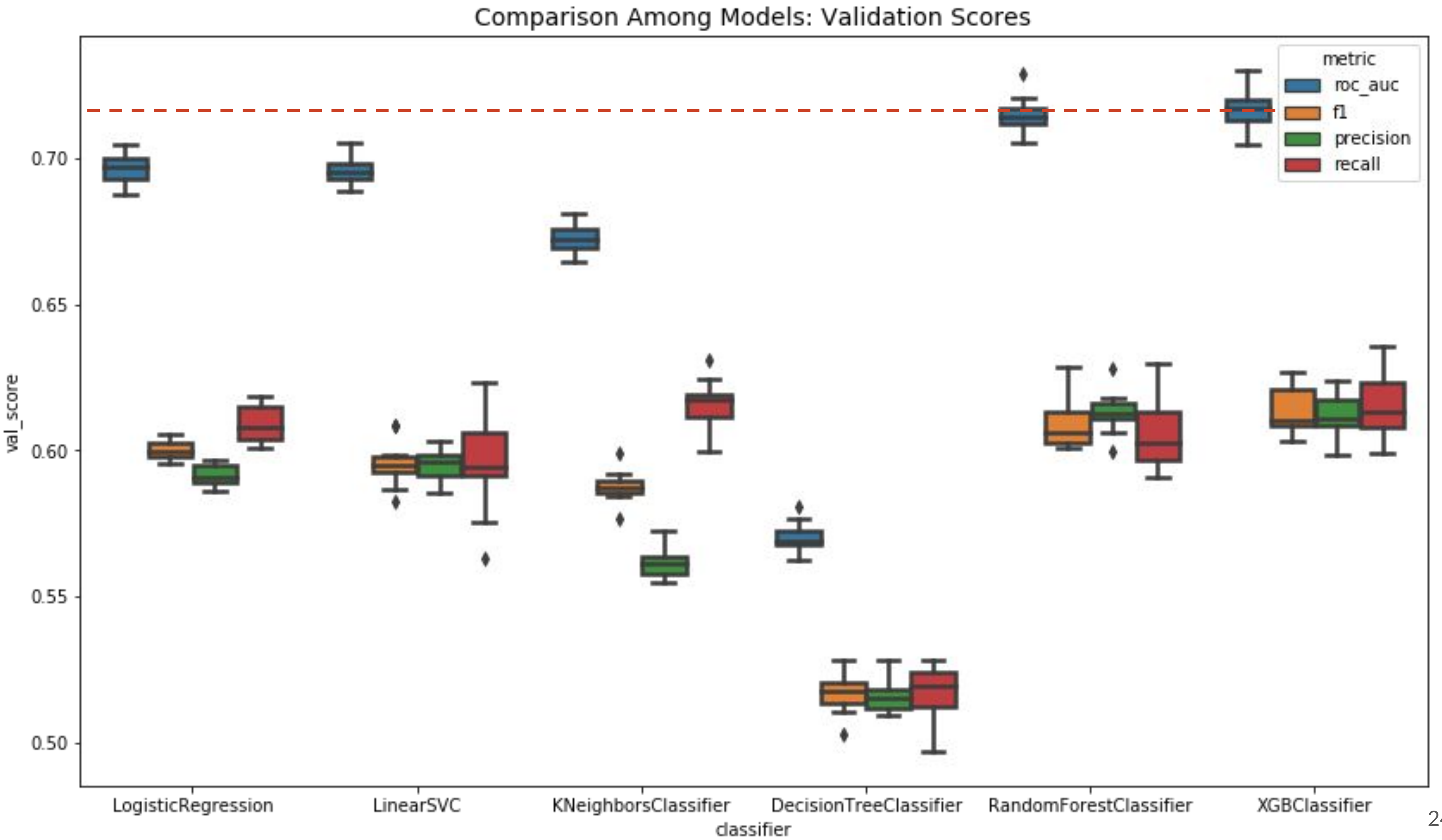
Hyperparameter Tuning



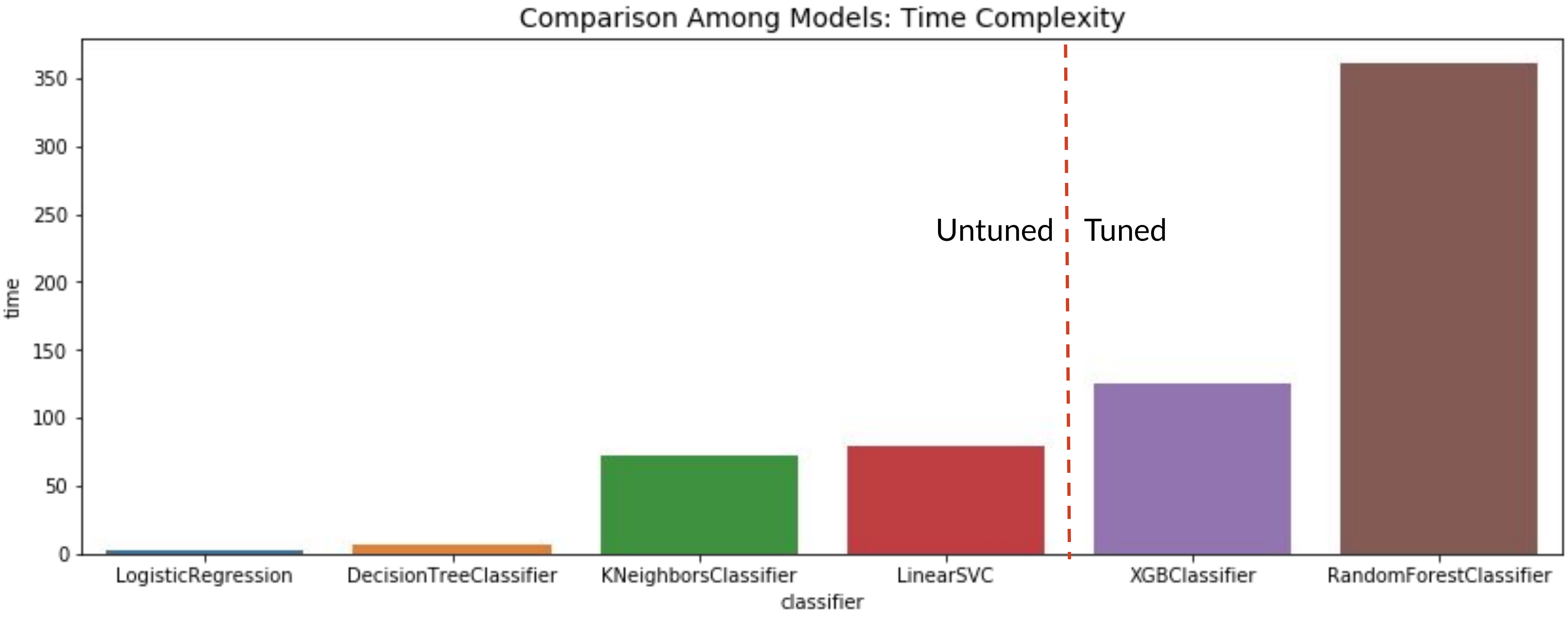
- Grid Search vs. Random Search
  - Same space of parameters, similar result in parameter setting
  - **Choice**: Random Search
    - Drastically lower runtime

Comparison Among Models

- Performance: validation scores
- Time complexity



Comparison Among Models: Validation Scores
metric
roc_auc
f1
precision
recall
val_score
0.70
0.65
0.60
0.55
0.50
LogisticRegression
LinearSVC
KNeighborsClassifier
DecisionTreeClassifier
RandomForestClassifier
XGBClassifier
classifier



16 vCPU, 60GB Memory

XGBoost.

But keep Logistic Regression in mind for its overall performance and simplicity.

Modeling:
Model Validation

XGBoost: Test Performance

$$\textbf{Precision} = \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Positives}}$$

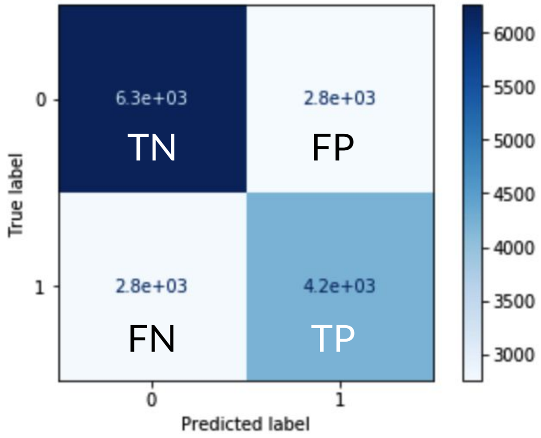
$$\textbf{Recall} = \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Negatives}}$$

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precision}^{-1}}{2}\right)^{-1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

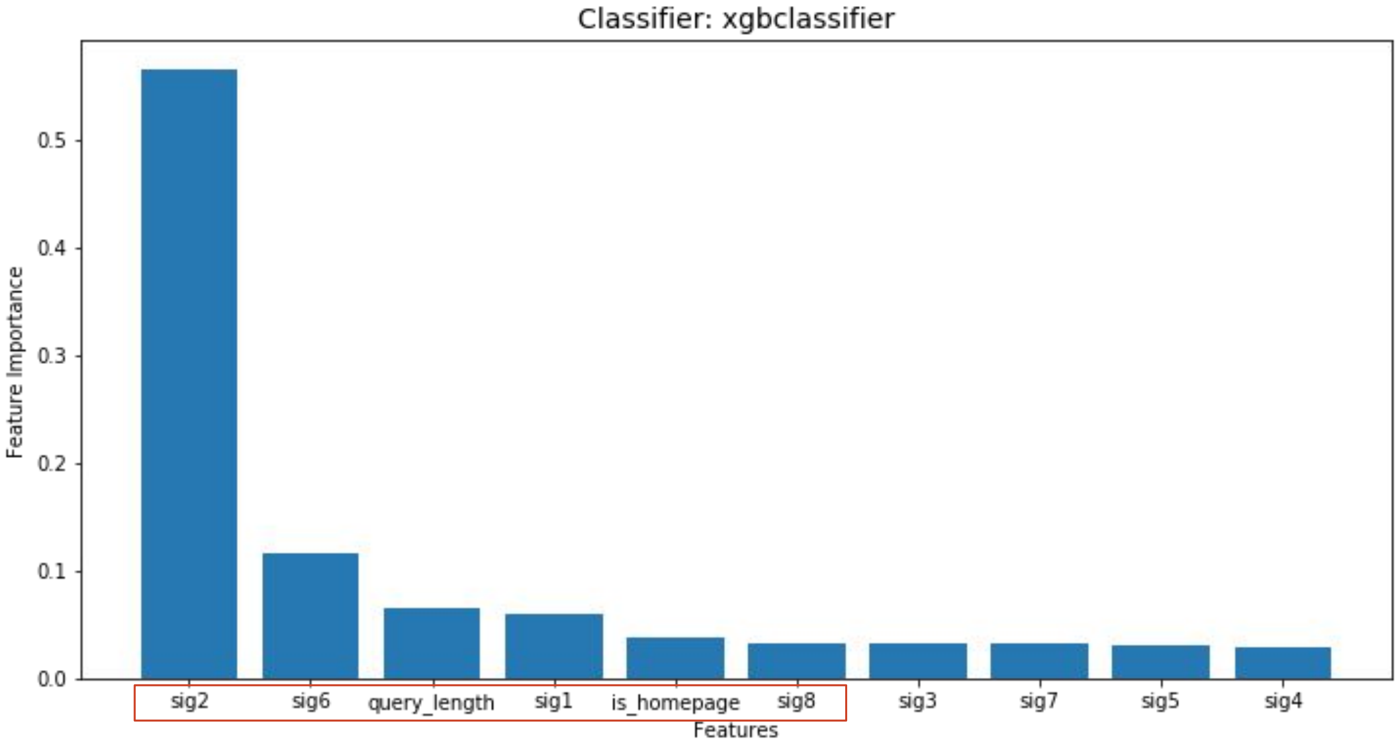
	precision	recall	f1-score	support
0	0.69	0.69	0.69	9012
1	0.61	0.60	0.60	6998
accuracy			0.65	16010
macro avg	0.65	0.65	0.65	16010
weighted avg	0.65	0.65	0.65	16010

```
Test ROC-AUC: 0.648766

Confusion matrix:
[[6255 2757]
 [2775 4223]]
```

Confusion Matrix for Test Data



Feature Importance



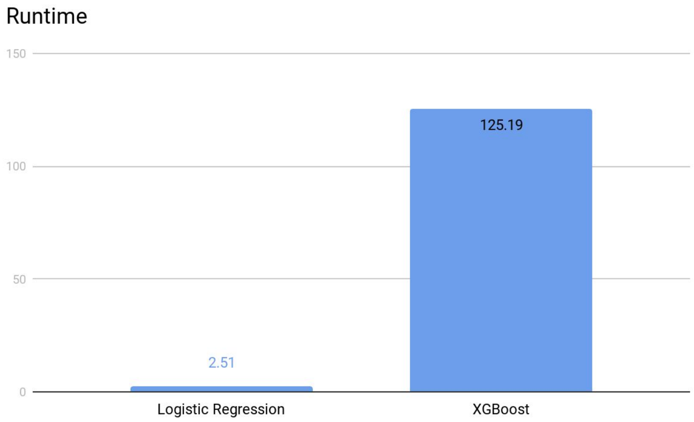
Business Insights & Recommendation

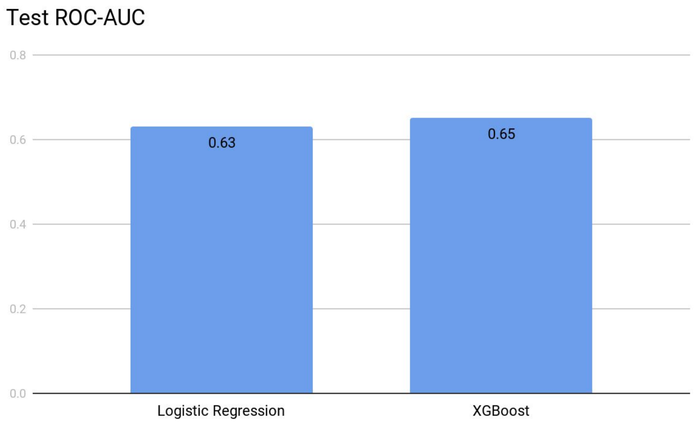
Explainable Model: Logistic Regression

Using important features identified by XGBoost:

```
'sig2', 'sig6', 'query_length',
'sig1', 'is_homepage', 'sig8'
```

```
Best parameters: {'logisticregression__C': 0.01}
Validation Scores:
      metric  val_score
3    roc_auc   0.692643
2     recall   0.604523
1  precision   0.587739
0         f1   0.595995
Execution Time: 2.51 s
```





Explainable Model: Logistic Regression

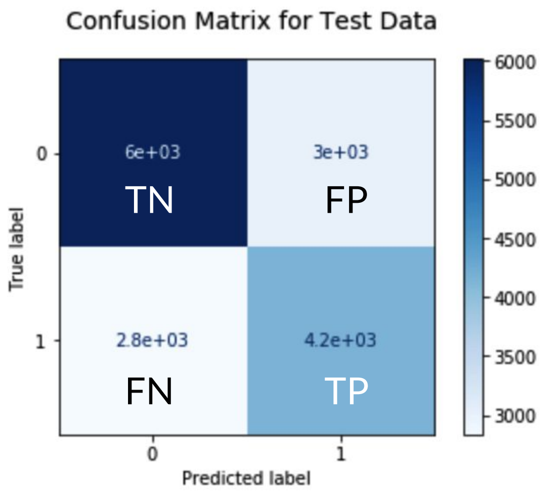
$$\textbf{Precision} = \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Positives}}$$

$$\textbf{Recall} = \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Negatives}}$$

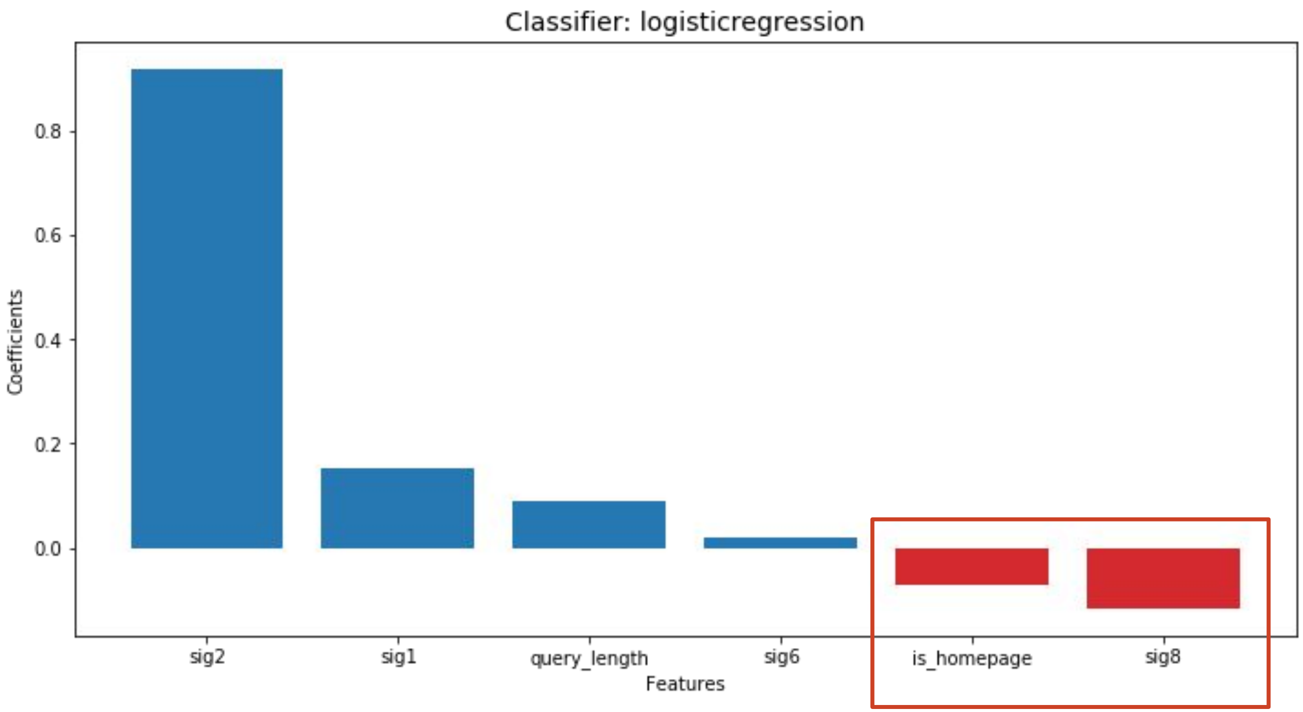
$$F_1 = \left(\frac{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}}{2}\right)^{-1} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

	precision	recall	f1-score	support
0	0.68	0.67	0.67	9012
1	0.58	0.59	0.59	6998
accuracy			0.64	16010
macro avg	0.63	0.63	0.63	16010
weighted avg	0.64	0.64	0.64	16010

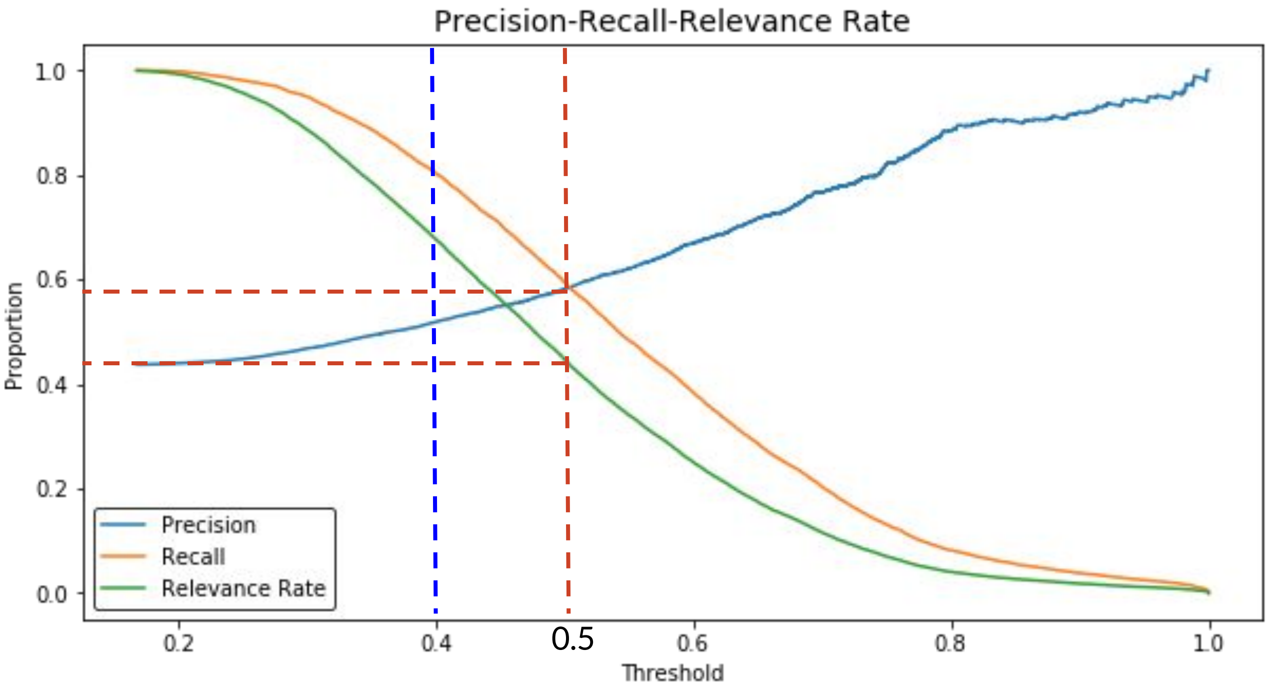
```
Test ROC-AUC: 0.631148

Confusion matrix:
[[6016 2996]
 [2836 4162]]
```



Coefficients: Direction of Correlation



Actionable Insights



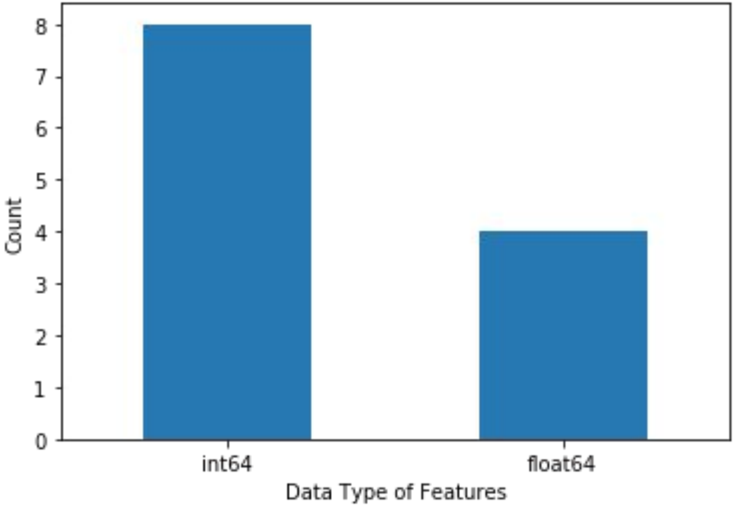
Beyond This Point

- Several improvements can be made
  - Analysis of the optimal threshold
  - Stratified cross-validation for more representative data
  - Fixing overfitting issues in XGBoost: e.g., early stopping
  - Advanced models such as Light GBM

Appendix:
EADS & Data Preprocessing

Data Types

Eight Discrete Features, Four Continuous Features



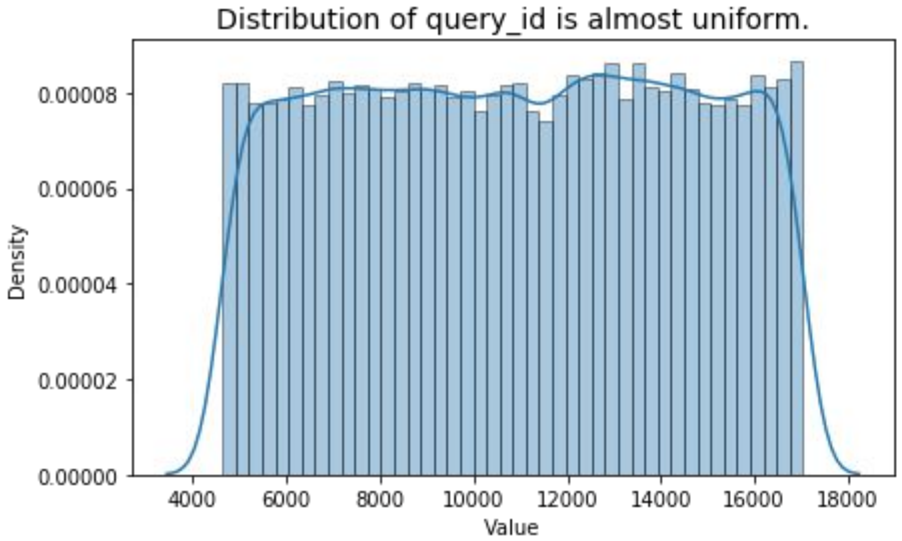

```
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   query_id      80046 non-null  int64
 1   url_id        80046 non-null  int64
 2   query_length  80046 non-null  int64
 3   is_homepage   80046 non-null  int64
 4   sig1          80046 non-null  float64
 5   sig2          80046 non-null  float64
 6   sig3          80046 non-null  int64
 7   sig4          80046 non-null  int64
 8   sig5          80046 non-null  int64
 9   sig6          80046 non-null  int64
 10  sig7          80046 non-null  float64
 11  sig8          80046 non-null  float64
 12  relevance     80046 non-null  int64
dtypes: float64(4), int64(9)
memory usage: 7.9 MB
```

Categorical

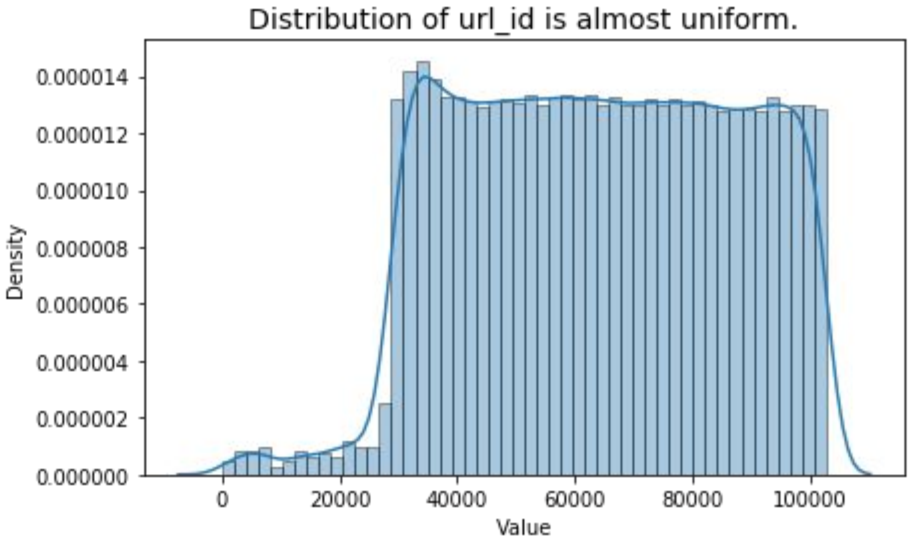
Exploratory Data Analysis: Univariate

Features

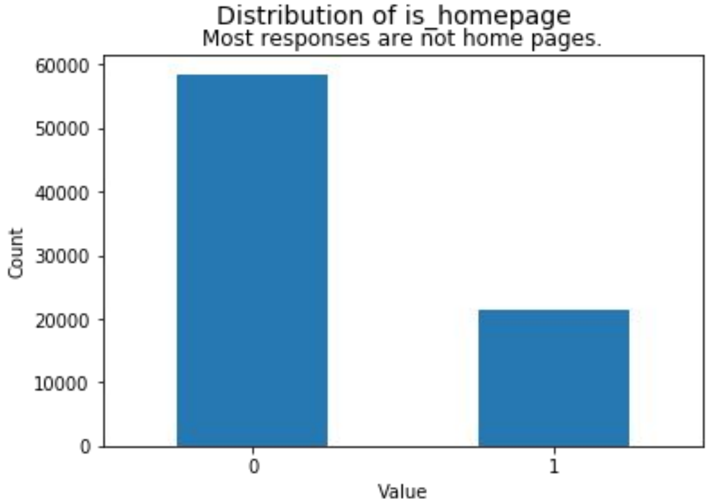
- 3 Categorical
  - `query_id, url_id`: dropped
  - `is_homepage`
- 4 Continuous
- 5 Discrete



Distribution of query_id is almost uniform.
Density
Value



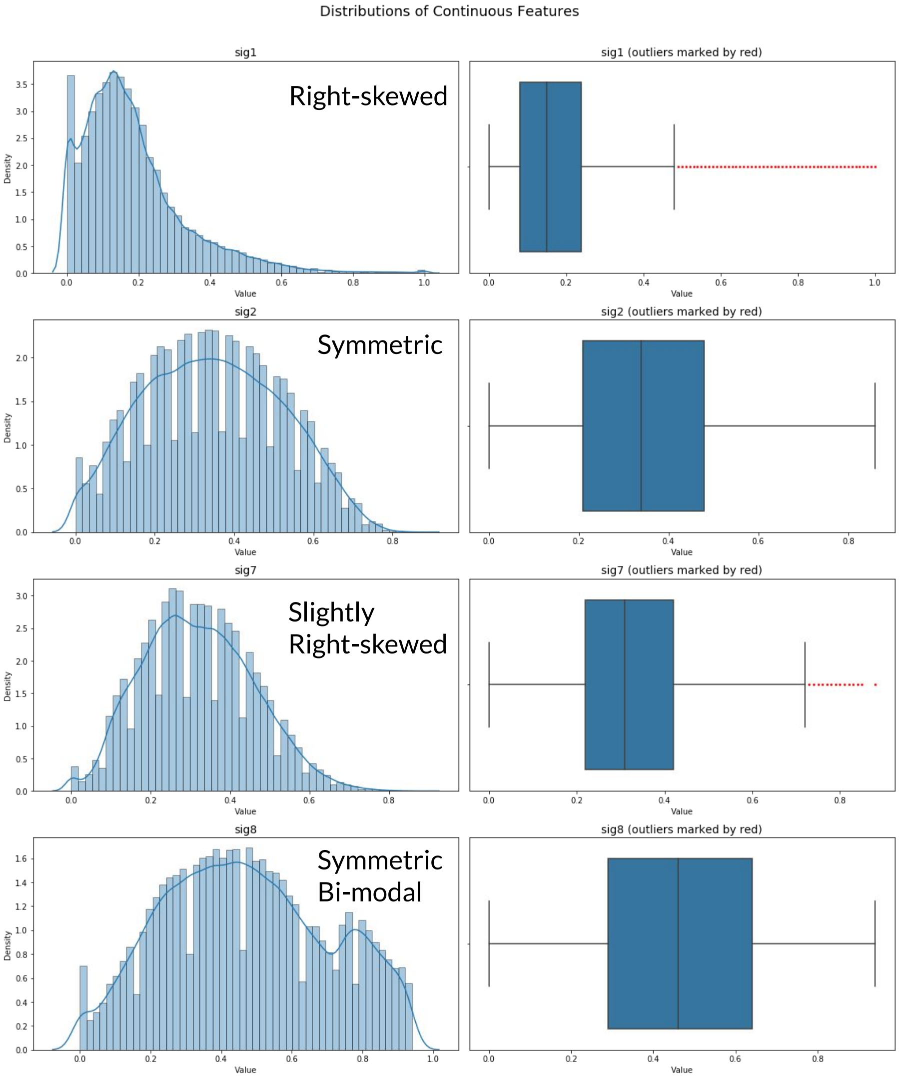
Distribution of url_id is almost uniform.
Density
Value



Distribution of is_homepage
Most responses are not home pages.
Count
Value

Features

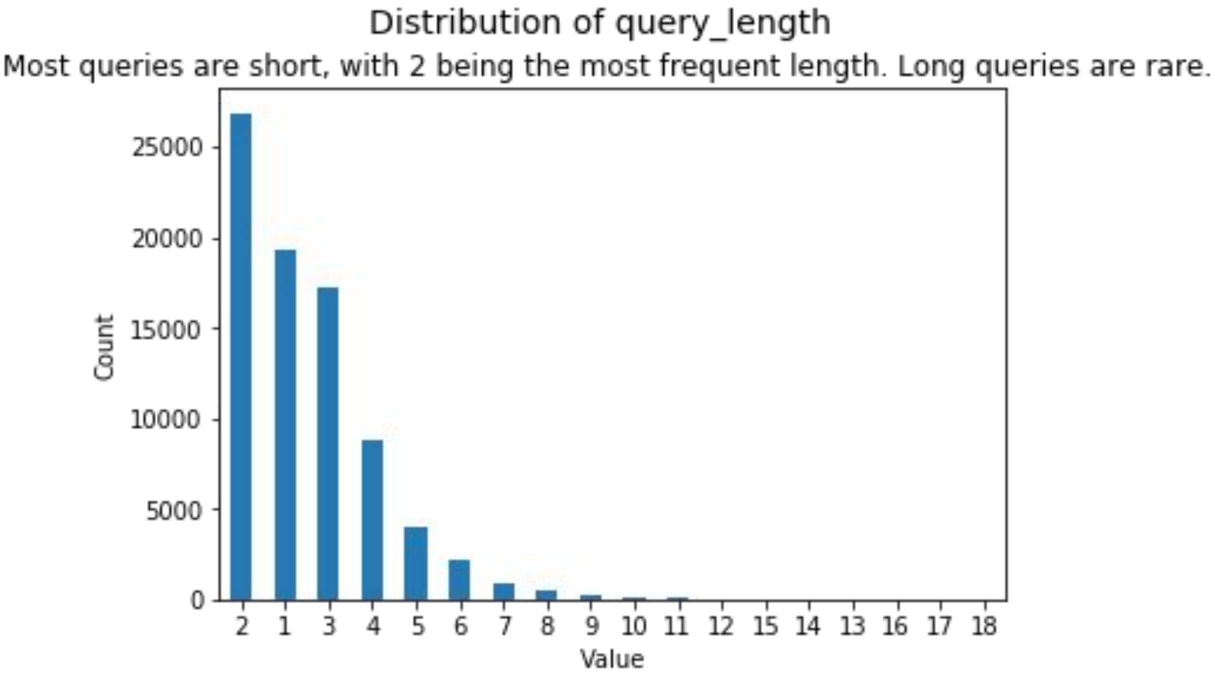
- 3 Categorical
- 4 Continuous: `sig1, sig2, sig7, sig8`
  - Range in (0, 1)
- 5 Discrete

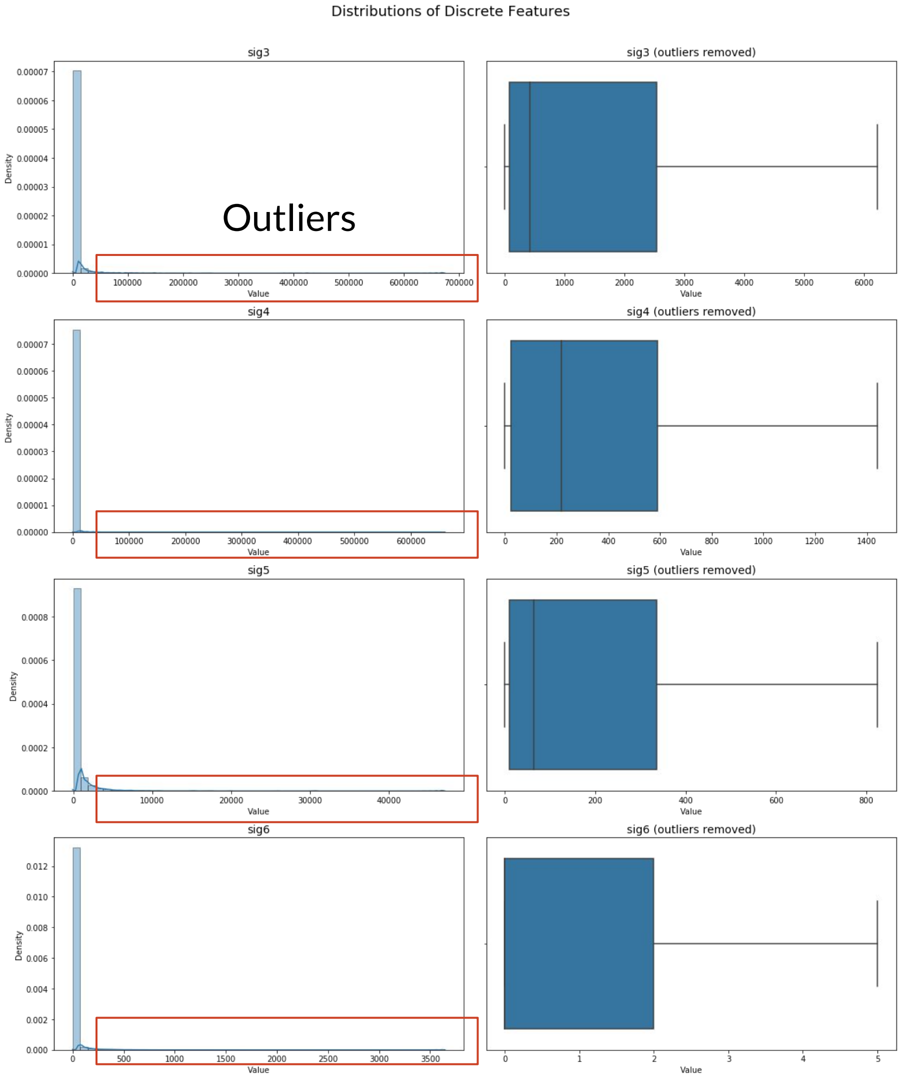


	sig1	sig2	sig7	sig8
count	80046.000000	80046.000000	80046.000000	80046.000000
mean	0.183240	0.346947	0.319464	0.471846
std	0.147354	0.172545	0.138651	0.231306
min	0.000000	0.000000	0.000000	0.000000
25%	0.080000	0.210000	0.220000	0.290000
50%	0.150000	0.340000	0.310000	0.460000
75%	0.240000	0.480000	0.420000	0.640000
max	1.000000	0.860000	0.880000	0.940000
mode	0.000000	0.320000	0.260000	0.470000

Features

- 3 Categorical
- 4 Continuous
- 5 Discrete:
  - `query_length`
    - Most queries are short (of length 1-5)
  - `sig3, sig4, sig5, sig6`
    - Varying scales
    - Outliers

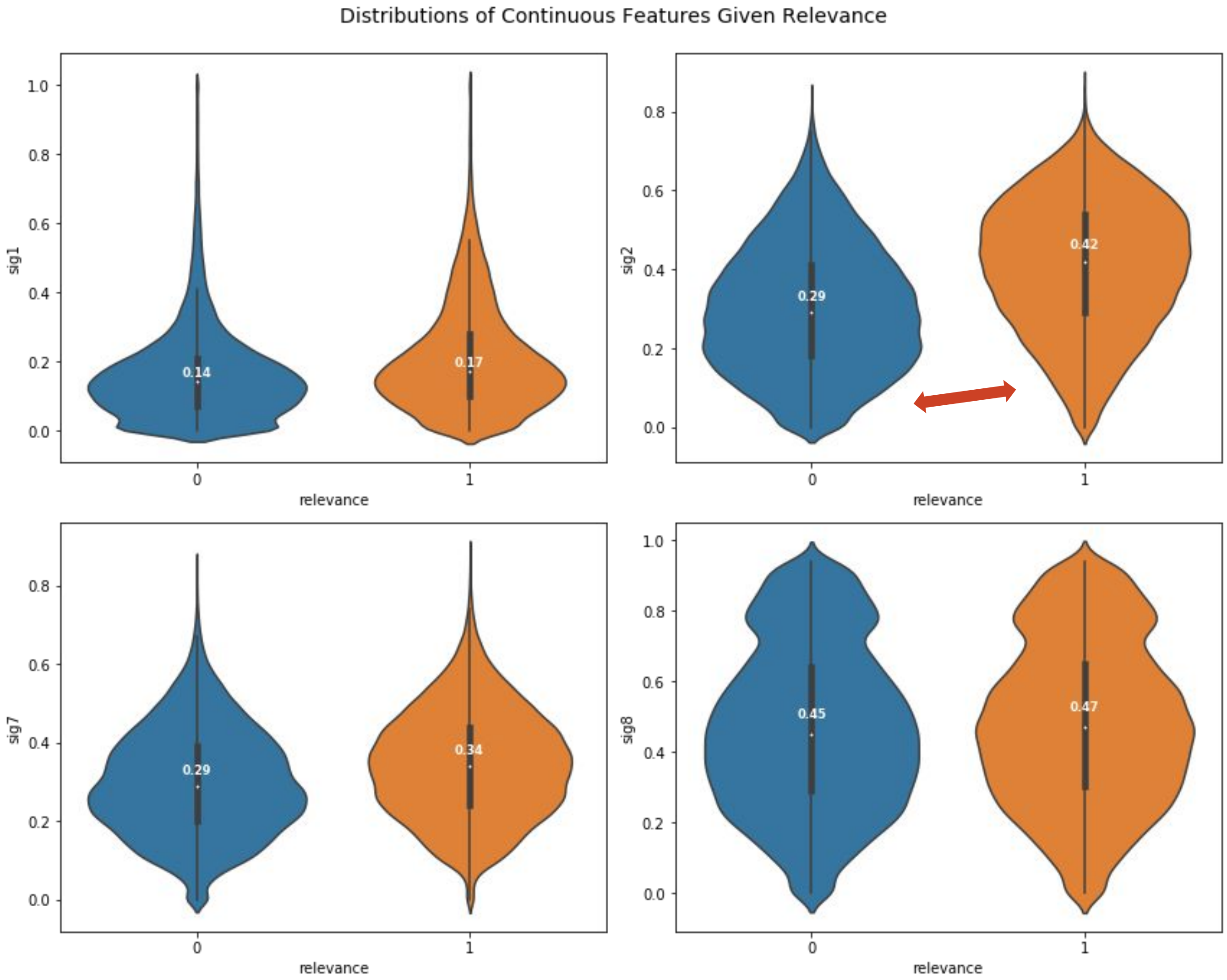

Distribution of query_length
Most queries are short, with 2 being the most frequent length. Long queries are rare.
25000
20000
15000
10000
5000
0
Count
2 1 3 4 5 6 7 8 9 10 11 12 15 14 13 16 17 18
Value



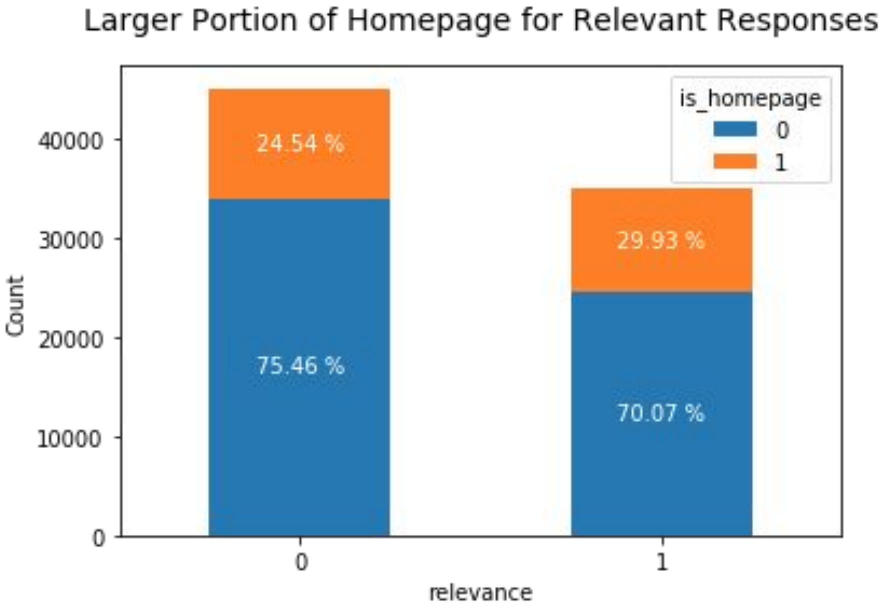
Scales vary a lot → need for scaling

	query_length	sig3	sig4	sig5	sig6
count	80046.000000	80046.000000	80046.000000	80046.000000	80046.000000
mean	2.585826	4857.078555	742.316256	550.527597	14.099155
std	1.522094	23531.973200	4818.359126	1887.933968	90.068426
min	1.000000	0.000000	0.000000	0.000000	0.000000
25%	2.000000	78.000000	24.000000	10.000000	0.000000
50%	2.000000	417.000000	220.000000	64.000000	0.000000
75%	3.000000	2537.750000	591.000000	336.000000	2.000000
max	18.000000	673637.000000	660939.000000	46994.000000	3645.000000
mode	2.000000	0.000000	0.000000	0.000000	0.000000

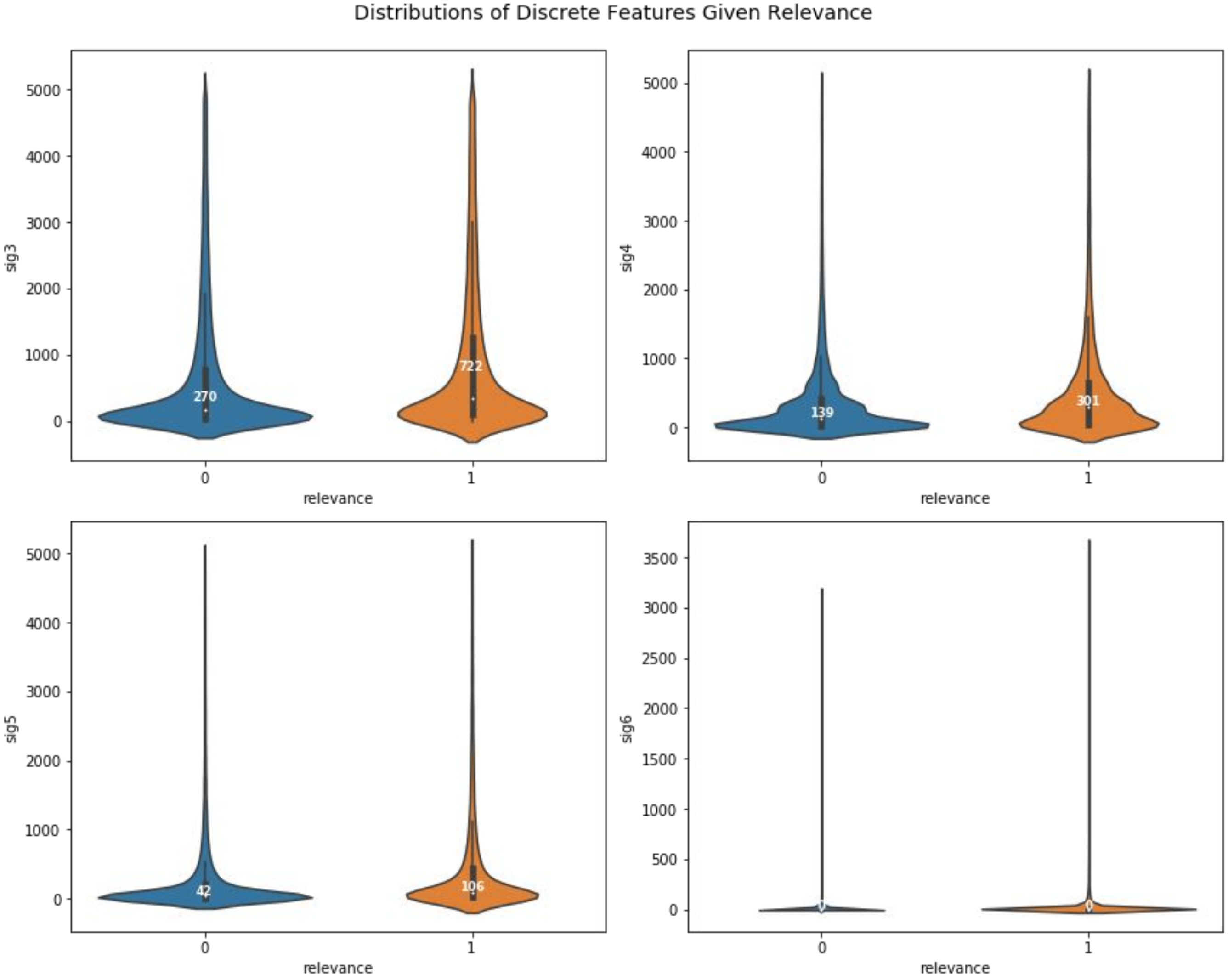
Exploratory Data Analysis: Multivariate



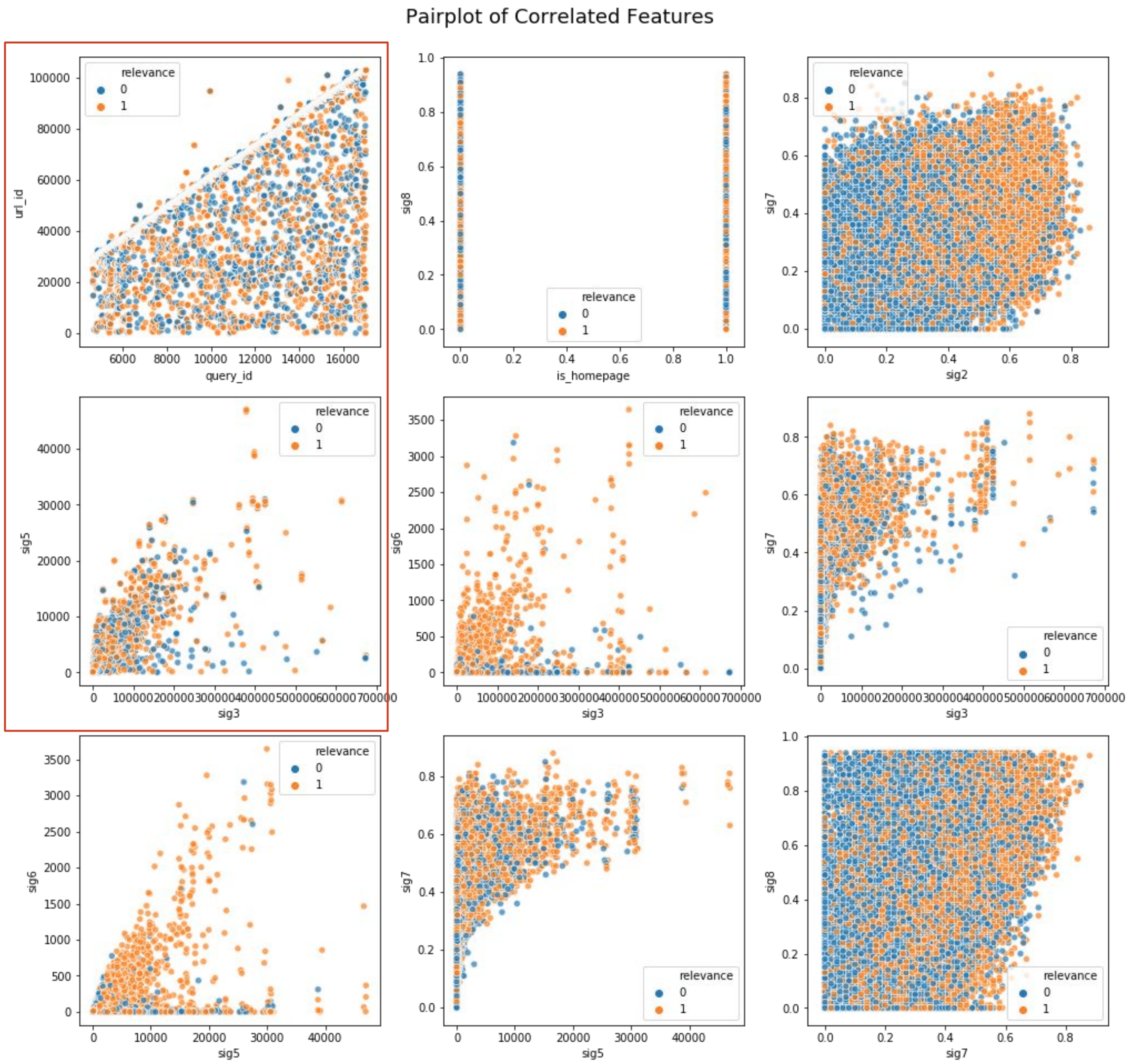
Distributions of `sig2` are different for different class of relevance (recall: high linear correlation of `relevance` vs. `sig2`



Relevance vs. is_homepage



Distributions of `relevance` for each feature at 0 vary

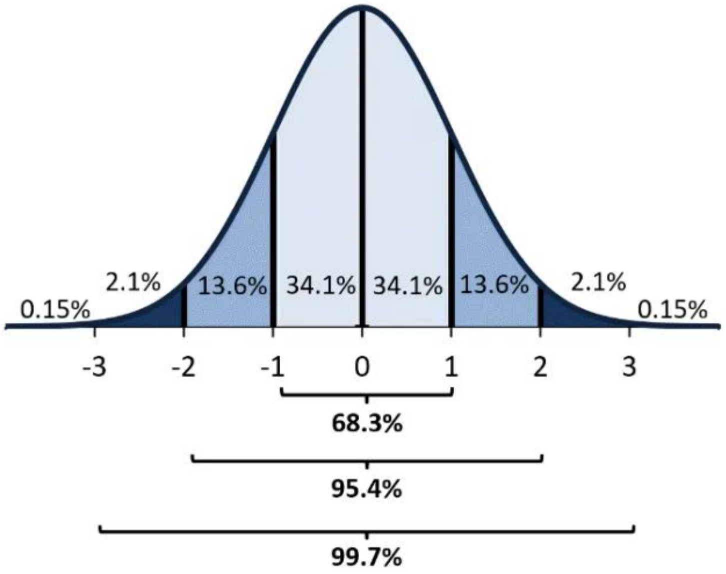


	feature_a	feature_b	corr_strength	corr
0	query_id	url_id	strong	0.91
3	sig3	sig5	strong	0.81
6	sig5	sig6	moderate	0.48
7	sig5	sig7	moderate	0.41
4	sig3	sig6	moderate	0.39
2	sig2	sig7	moderate	0.37
8	sig7	sig8	moderate	0.32
5	sig3	sig7	moderate	0.30
1	is_homepage	sig8	moderate	-0.34

- Caveat of multicollinearity: Unstable coefficient estimates → difficult to interpret
- Decide whether to remove one of the pair to avoid redundant information.

Outliers

- Percentage of outliers based on Z-score (Z > 3): 5.18 %
- Many of the outliers are from the unlabelled features which we do not have knowledge of
    - Such outliers are not removed.



$$z = \frac{x - \mu}{\sigma}$$

$\mu =$ Mean

$\sigma =$ Standard Deviation

Clarification: Correlation Heatmap

- Method: Pearson
- With two levels of the target, Point Biserial Correlation is equivalent to the Pearson Correlation

	Categorical	Continuous
Categorical	Lambda, Corrected Cramer's V	Point Biserial, Logistic Regression
Continuous	Point Biserial, Logistic Regression	Spearman, Kendall, Pearson

Feature Engineering: Removed `query_id, url_id`

Univariate F-test

- Univariate feature selection works by selecting the best features based on univariate statistical tests
- F-test: determines whether the variability between group means is larger than the variability of the observations within the groups
- **Ideal**: large F-statistic
- At significance level of 0.1, `query_id` and `query_length` are not significant.
- `url_id` also has a low importance in terms of F-score.

$$F = \frac{\text{between-groups variance}}{\text{within-group variance}}$$

	feature	F_score	p_value
5	sig2	8258.282169	0.000000e+00
10	sig7	2244.136829	0.000000e+00
4	sig1	2108.304535	0.000000e+00
9	sig6	1260.511738	5.900848e-274
8	sig5	862.084350	1.728422e-188
6	sig3	425.818839	2.328943e-94
3	is_homepage	291.993649	2.389578e-65
7	sig4	94.034436	3.189444e-22
11	sig8	66.126484	4.288830e-16
1	url_id	37.391156	9.710474e-10
0	query_id	0.007469	9.311294e-01
2	query_length	0.000236	9.877413e-01

L1-based model approach

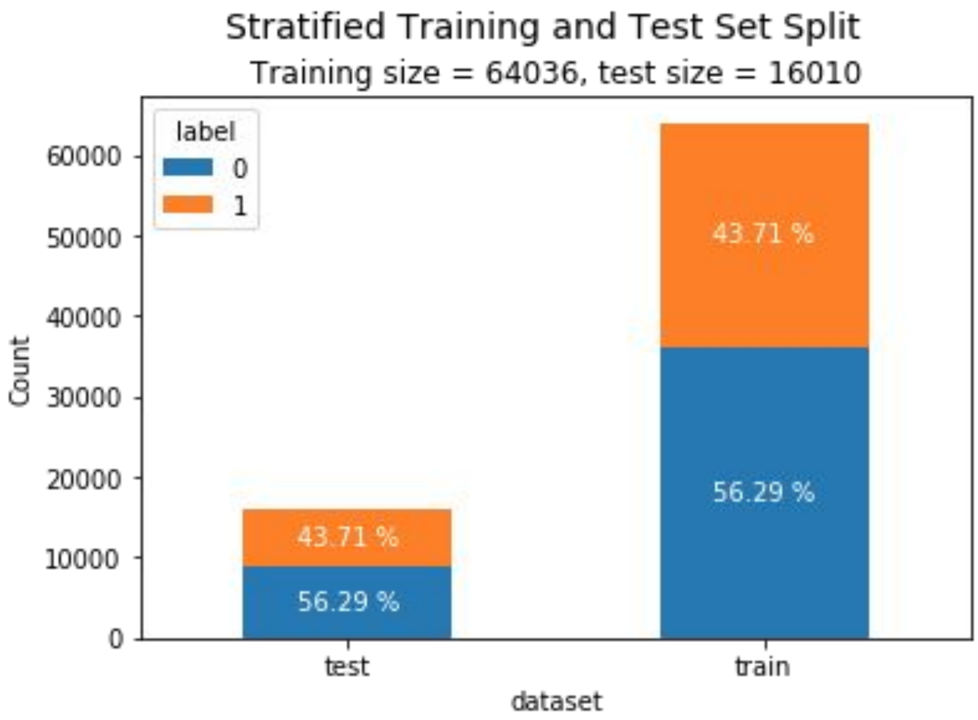
- Idea: sparse solution removes estimated 0 coefficients
- `LinearSVC(C=0.01, penalty='l1')`
- Insignificant features: `query_id`, `url_id`, `sig3`, `sig4` and `sig5`

Data Segregation: Train-test-split

Stratified Train-test-split

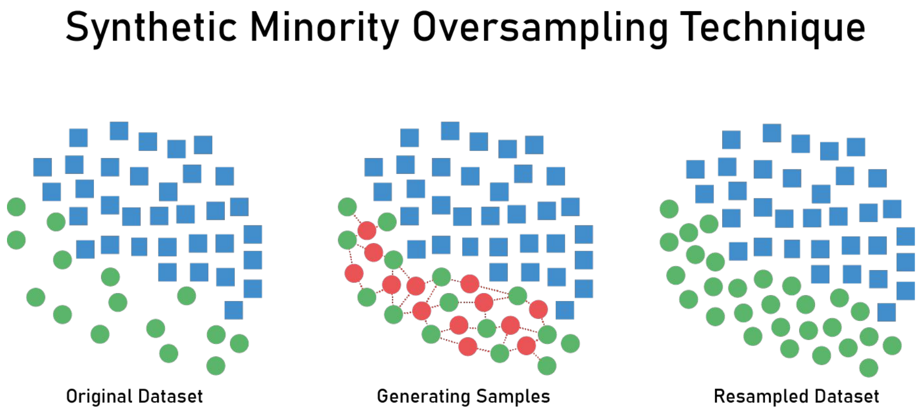
Preserves the proportion of target as in original dataset, in the train and test datasets

→ Reproducibility, better prediction



SMOTE

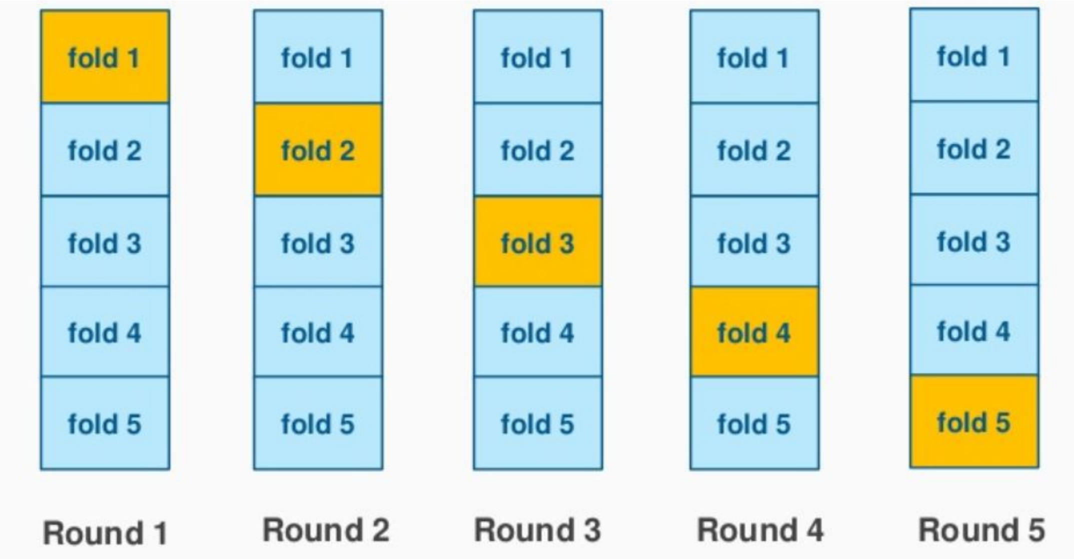
- Idea
  - Select data points that are close in the feature space
  - Draw a line between the points in the feature space
  - Draw a new sample at a point along that line
- Pros
  - Synthetic samples mitigate the problem of overfitting
  - No loss of useful information
- Cons
  - Not for high dimensional data
  - Adds noise



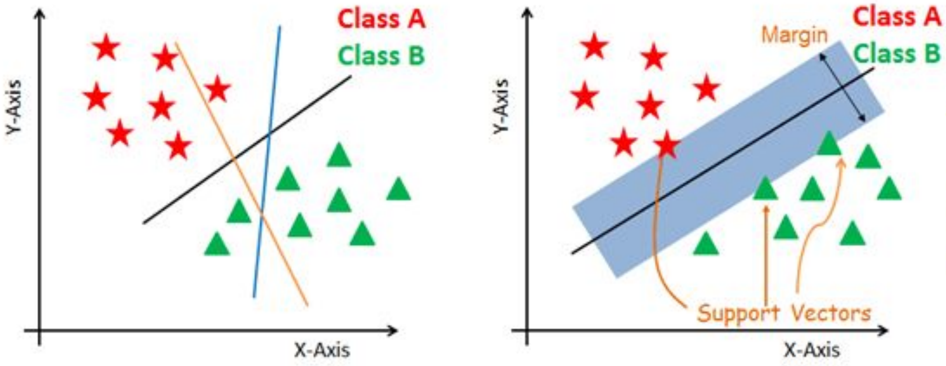
Appendix: Modeling

K-fold Cross-validation

- Partition the data into k subsets
- Iteratively train the algorithm on the k-1 folds while using the remaining fold as the test set
- Prevents overfitting



Linear SVM



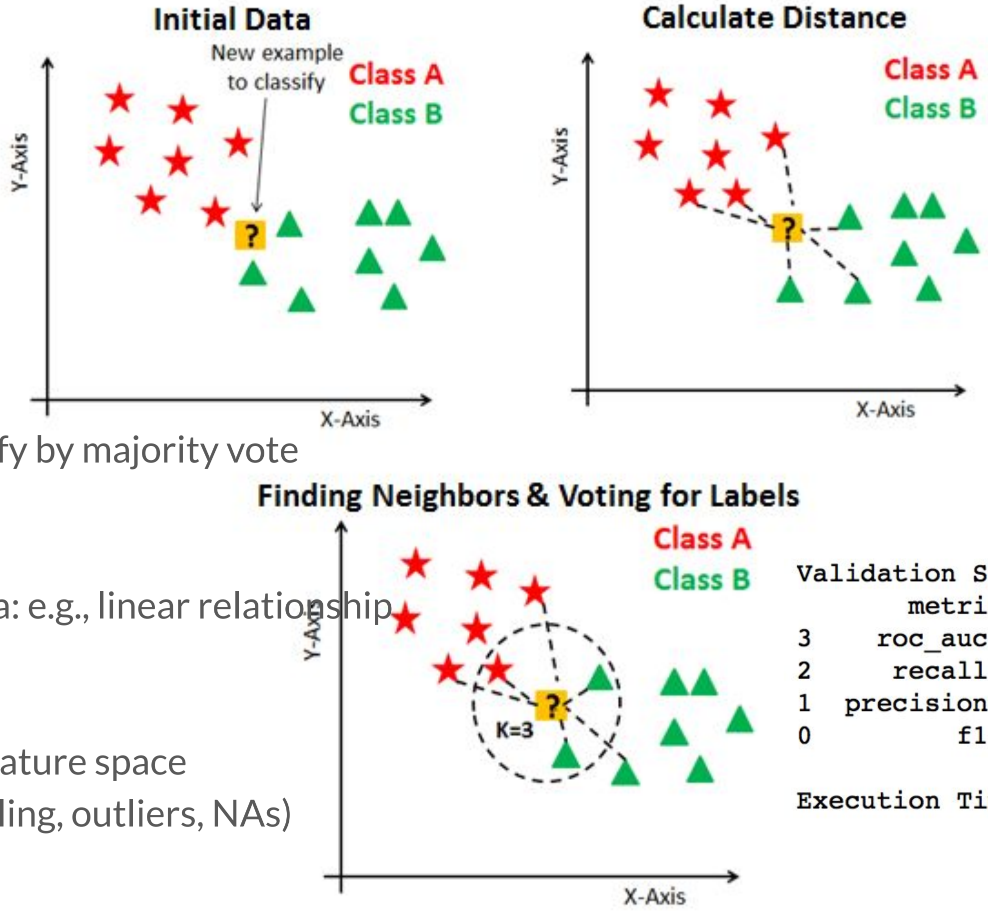

```
Validation Scores:
          metric  val_score
3        roc_auc   0.695940
2         recall   0.596557
1      precision   0.595081
0             f1   0.595634

Execution Time: 79.36 s
```

- Find a hyperplane that leaves the maximum margin from both classes
- Pros
  - Good for linearly separable data
  - Good for large feature space
  - Less likely to overfit
- Cons
  - Slow for large datasets
  - Needs careful tuning
  - Needs scaling

A note on non-linear kernels:

- I did not experiment with non-linear kernels for **time complexity**. The runtime for a non-linear kernel is roughly $O(n\_samples^{\wedge}2 * n\_features)$. Since we have 64000 data points along with 10 features, a non-linear kernel can explode runtime, let alone the time complexity introduced by cross-validation or grid search.

K Nearest Neighbors (KNN)

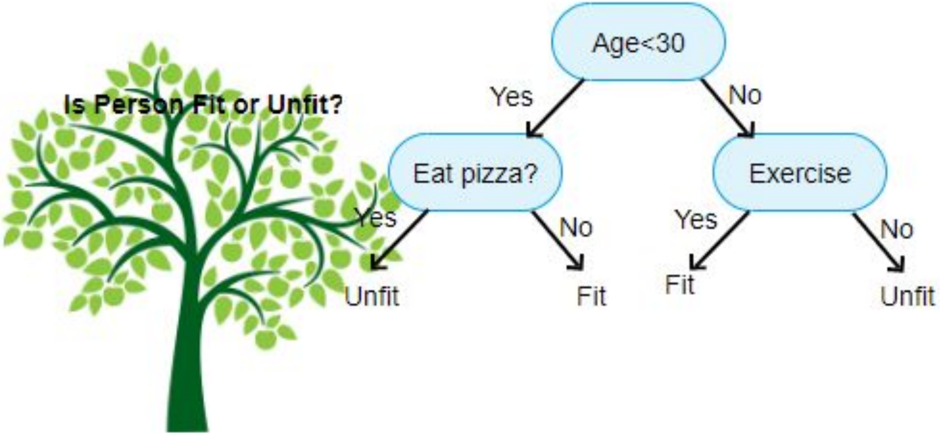


- Find K nearest points and classify by majority vote
- Pros
  - Simple
  - No assumption about data: e.g., linear relationship
  - Adapts to new data
- Cons
  - Slow for large datasets/feature space
  - Needs preprocessing (scaling, outliers, NAs)

```
Validation Scores:
       metric  val_score
3     roc_auc   0.672587
2      recall   0.615813
1   precision   0.562095
0          f1   0.587696

Execution Time: 72.98 s
```

Decision Tree

- Pros
  - Interpretable
  - Business insights
  - Resistant to outliers, hence require little data preprocessing
- Cons
  - Prone to overfitting
  - Instability



GINI Impurity: $I_G(n) = 1 - \sum_{i=1}^{J} (p_i)^2$

```
Validation Scores:
        metric  val_score
3      roc_auc   0.570472
2       recall   0.516953
1    precision   0.516348
0           f1   0.516613

Execution Time: 6.31 s
```

Evolution from Decision Trees