

Machine Learning Systems Design

Data Lifecycle

Lecture 8: Feature Engineering



CE 40959 Spring 2023

Ali Zarezade

[SharifMLSD.github.io](https://github.com/SharifMLSD)

Agenda

1. Why Feature Engineering
2. Handling Missing Values
3. Feature Scaling and Transformation
4. Feature Selection

1. Why Feature Engineering

Feature engineering

In 2014, the paper “Practical Lessons from Predicting Clicks on Ads at Facebook” claimed that having the right features is the most important thing in developing their ML models.

Feature engineering

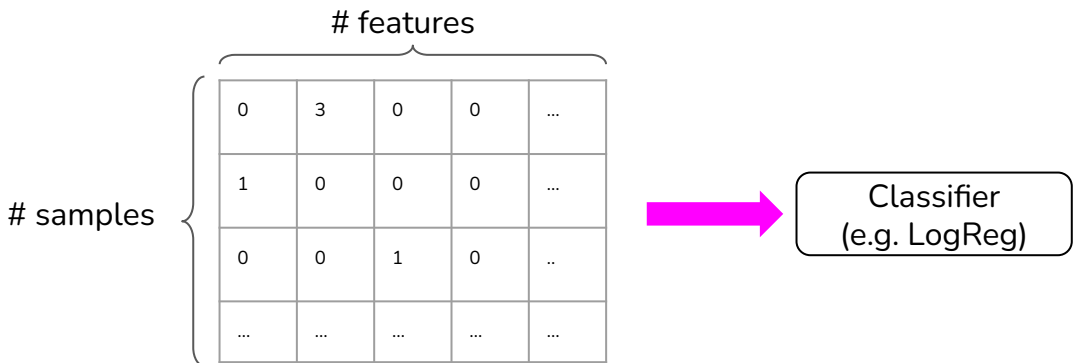
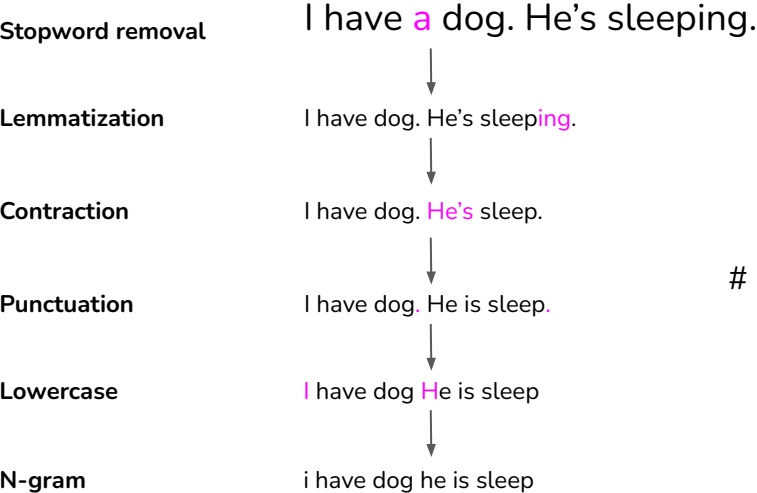
Wait, doesn't deep learning promise no more feature engineering?

Feature engineering

Wait, doesn't deep learning promise no more feature engineering?

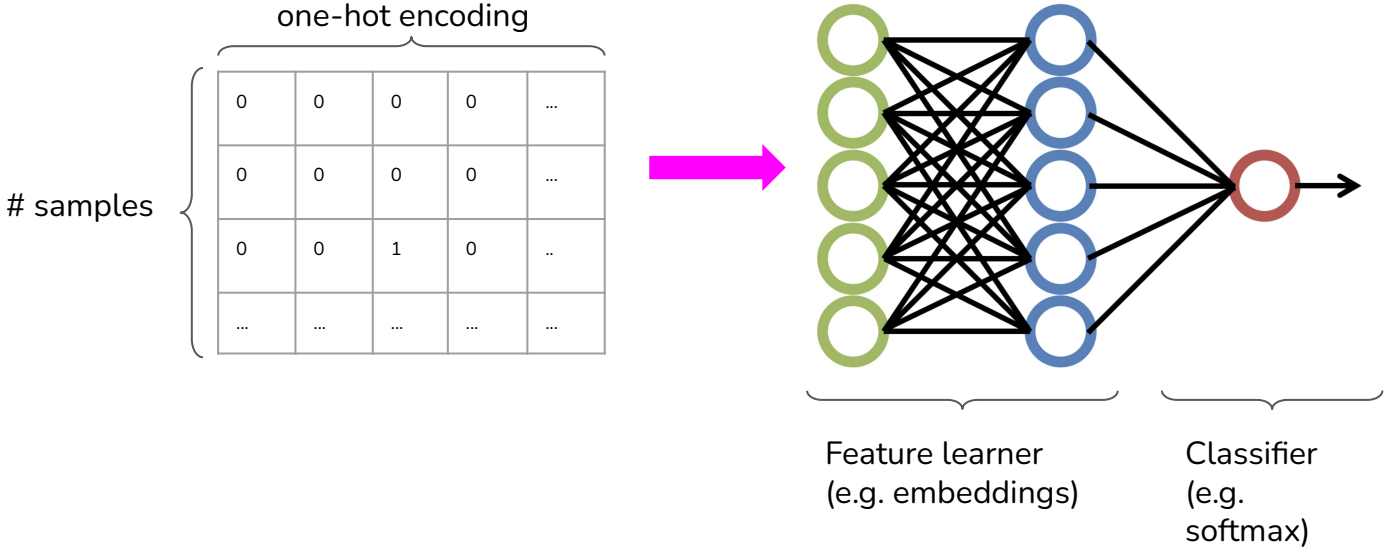
- We're still very far from that point
- Many ML models in industry aren't deep learning
- Right features give performance boost

Engineered features: text



Features	I	you	have	dog	cat	he	she	is	they	sleep	I, have	have, dog	good, dog	...
	1	0	1	1	0	1	0	1	0	1	1	1	0	...

Learned features: text







Text I have a dog. He's sleeping.



One-hot

I	you	have	dog	cat	he	she	is	they	sleep	mom	food	yes	...
1	0	1	1	0	1	0	1	0	0	0	0	0	...

Learned features: spam classification

Comment ID	Time	User	Text	# 	# 	Link	# img	Thread ID	Reply to	# replies	...
93880839	2020-10-30 T 10:45 UTC	gitrekt	Your mom is a nice lady.	1	0	0	0	2332332	n0tab0t	1	...

User ID	Created	User	Subs	# 	# 	# replies	Karma	# threads	Verified email	Awards	...
4402903	2015-01-57 T 3:09 PST	gitrekt	[r/ml, r/memes, r/socialist]	15	90	28	304	776	No		...

Thread ID	Time	User	Text	# 	# 	Link	# img	# replies	# views	Awards	...
93883208	2020-10-30 T 2:45 PST	doge	Human is temporary, AGI is forever	120	50	1	0	32	2405	1	...

Some of the possible features about a comment, a thread, or a user to be included in your model

Feature engineering: spam classification

Even more features:

- Post frequency, max posts per day
- Post repetitiveness
- Language detection, typos, abnormal punctuations, ratio uppercase/lowercase
- IP, other users from the same IP
- NSFW words, blacklisted links
- Targeted users
- ...

Feature engineering

- For complex tasks, number of features can go up to millions! (TikTok videos)
- Lots of ML production work involves coming up with new features
 - Fraudsters come up with new techniques very fast, so need to come up with new features very fast to counter
- Often require subject matter expertise (SMEs)

2. Handling Missing Values

Handling missing values

- Not all missing values are equal
 - Missing not at random (MNAR)
 - Missing at random (MAR)
 - Missing completely at random (MCAR)



Handling missing values

Example data for predicting house buying in the next 12 months

ID	Age	Gender	Annual income	Marital status	Number of children	Job	Buy?
1		A	150,000		1	Engineer	No
2	27	B	50,000			Teacher	No
3		A	100,000	Married	2		Yes
4	40	B			2	Engineer	Yes
5	35	B		Single	0	Doctor	Yes
6		A	50,000		0	Teacher	No
7	33	B	60,000	Single		Teacher	No
8	20	B	10,000			Student	No

Handling missing values

Missing not at random – when a value is missing due to the value itself

ID	Age	Gender	Annual income	Marital status	Number of children	Job	Buy?
1		A	150,000		1	Engineer	No
2	27	B	50,000			Teacher	No
3		A	100,000	Married	2		Yes
4	40	B	(\$350,000?)		2	Engineer	Yes
5	35	B	(\$350,000?)	Single	0	Doctor	Yes
6		A	50,000		0	Teacher	No
7	33	B	60,000	Single		Teacher	No
8	20	B	10,000			Student	No

Handling missing values

Missing at random – when a value is missing due to another observed variable

ID	Age	Gender	Annual income	Marital status	Number of children	Job	Buy?
1		A	150,000		1	Engineer	No
2	27	B	50,000			Teacher	No
3		A	100,000	Married	2		Yes
4	40	B			2	Engineer	Yes
5	35	B		Single	0	Doctor	Yes
6		A	50,000		0	Teacher	No
7	33	B	60,000	Single		Teacher	No
8	20	B	10,000			Student	No

Handling missing values

Missing completely at random – there is no pattern to which values are missing

ID	Age	Gender	Annual income	Marital status	Number of children	Job	Buy?
1		A	150,000		1	Engineer	No
2	27	B	50,000			Teacher	No
3		A	100,000	Married	2		Yes
4	40	B			2	Engineer	Yes
5	35	B		Single	0	Doctor	Yes
6		A	50,000		0	Teacher	No
7	33	B	60,000	Single		Teacher	No
8	20	B	10,000			Student	No

Handling missing values

- Deletion – removing data with missing entries
- Imputation – filling missing fields with certain values

Many people prefer deletion not because it's better, but it's easier to do

Handling missing values

- Deletion
 - Column deletion – remove columns with too many missing entries
 - drawbacks – even if half the values are missing, the remaining data still potentially useful information for predictions
 - e.g. even if over half the column for ‘Marital status’ is missing, marital status is still highly correlated with house purchasing
 - Row deletion

Marital status
Married
Single
Single

Handling missing values

- Deletion
 - Column deletion
 - Row deletion

Handling missing values

- Row deletion
 - Good for: data missing completely at random (MCAR) and few values missing

ID	Age	Gender	Annual income	Marital status	Number of children	Job	Buy?
1	39	A	150,000	Married	1	Engineer	No
2	27	B	50,000	Single	0	Teacher	No
3		A	100,000	Married	2		Yes
4	40	B	75,000	Married	2	Engineer	Yes
5	35	B	35,000	Single	0	Doctor	Yes
6	32	A	50,000	Married	0	Teacher	No
7	33	B	60,000	Single	2	Teacher	No
8	20	B	10,000	Single	1	Student	No

Handling missing values

- Row deletion
 - Bad when many examples have missing fields

ID	Age	Gender	Annual income	Marital status	Number of children	Job	Buy?
1		A	150,000		1	Engineer	No
2	27	B	50,000			Teacher	No
3		A	100,000	Married	2		Yes
4	40	B			2	Engineer	Yes
5	35	B		Single	0	Doctor	Yes
6		A	50,000		0	Teacher	No
7	33	B	60,000	Single		Teacher	No
8	20	B	10,000			Student	No

Handling missing values

- Row deletion
 - Bad for: missing values are not at random (MNAR)
 - Missing information is information itself

ID	Age	Gender	Annual income	Marital status	Number of children	Job	Buy?
1		A	150,000		1	Engineer	No
2	27	B	50,000			Teacher	No
3		A	100,000	Married	2		Yes
4	40	B	(\$350,000?)		2	Engineer	Yes
5	35	B	(\$350,000?)	Single	0	Doctor	Yes
6		A	50,000		0	Teacher	No
7	33	B	60,000	Single		Teacher	No
8	20	B	10,000			Student	No

Handling missing values

- Row deletion
 - Bad for: missing data at random (MAR)
 - Can potentially bias data – we've accidentally removed all examples with gender 'A'

ID	Age	Gender	Annual income	Marital status	Number of children	Job	Buy?
1		A	150,000		1	Engineer	No
2	27	B	50,000			Teacher	No
3		A	100,000	Married	2		Yes
4	40	B			2	Engineer	Yes
5	35	B		Single	0	Doctor	Yes
6		A	50,000		0	Teacher	No
7	33	B	60,000	Single		Teacher	No
8	20	B	10,000			Student	No

Imputation

- Using information and relationships among the nonmissing features to provide an estimate to fill in the missing value.
- With imputation, you risk injecting your own bias into and adding noise to your data, or worse, data leakage

Imputation

- Defaults
 - e.g. 0, or the empty string, etc.
 - avoid filling missing values with possible values! (e.g. number of children with default 0!)

Imputation

- Defaults
 - e.g. 0, or the empty string, etc.
 - avoid filling missing values with possible values! (e.g. number of children with default 0!)
- Statistical measures – mean, median, mode
 - e.g. if a day in July is missing its temperature value, fill it with the median temperature in July

Imputation

- Defaults
 - e.g. 0, or the empty string, etc.
 - avoid filling missing values with possible values! (e.g. number of children with default 0!)
- Statistical measures – mean, median, mode
 - e.g. if a day in July is missing its temperature value, fill it with the median temperature in July
- Model based - KNN, trees
 - identifies the K most similar samples in the training data that are complete
 - e.g., if you want to impute the missing values for a variable x1, you can train a decision tree using all other variables as predictors and x1 as the target. The predicted value of x1 is then used as the imputed value.

3. Feature Scaling and Transformation

Why feature scaling

Why feature scaling is important?

Why feature scaling and transformation

Why feature scaling is important?

- Performance boost especially in classical algorithms.
- It can help improve the performance and accuracy of some machine learning algorithms that calculate distances between data (K-means clustering).
- Performance boost in problems with small dataset size.
- It can help meet some assumptions for certain statistical methods that require normality.
- Reduce the effect of outliers on the model.
- Improving the convergence speed and accuracy of gradient descent.

Feature scaling vs transformation

- **Feature scaling:** rescaling each feature such that it has a similar range or magnitude, which can improve the performance of algorithms that are sensitive to the scale of features (**linear, dist. preserving**).
- **Feature transformation:** changing the shape or distribution of each feature, which can help to reduce skewness, outliers, or non-linearity (**nonlinear, dist. changes**).

Feature scaling

Common feature scaling methods:

- **Min-Max scaling:** rescales features using minimum and maximum values.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \quad \text{no assumption about variables}$$

Feature scaling

Common feature scaling methods:

- Min-Max scaling: rescales features using minimum and maximum values.
- **Standardization**: rescales features to have a mean of 0 and a standard deviation of 1.

$$x' = \frac{x - \bar{x}}{\sigma}$$

When variables follow a normal distribution

Feature scaling

Common feature scaling methods:

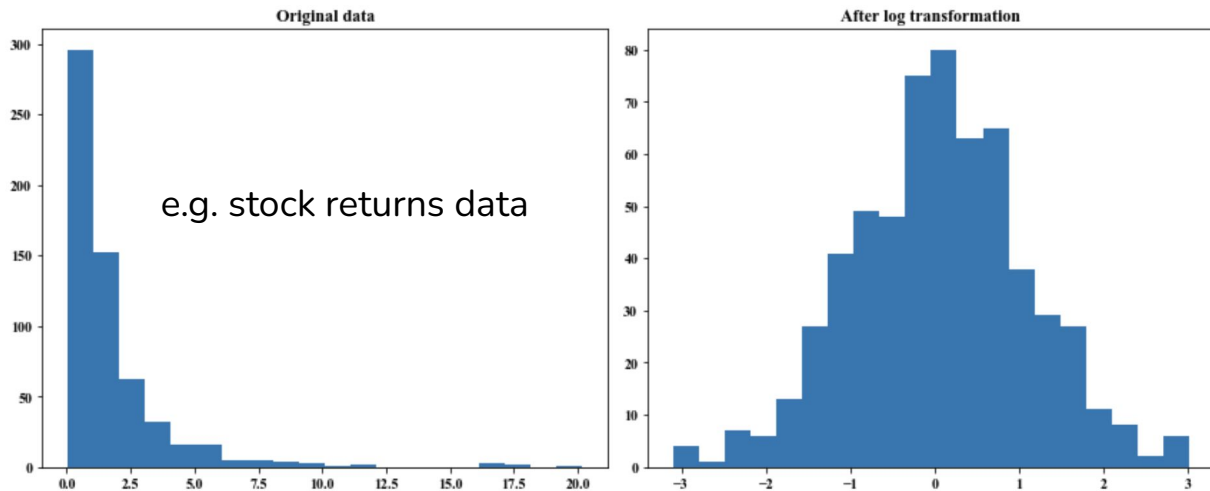
- Min-Max scaling: rescales features using minimum and maximum values.
- Standardization: rescales features to have a mean of 0 and a standard deviation of 1.
- **Robust Scaler**: It scales the data according to the quantile range (robust to outliers)

$$X_{\text{scale}} = \frac{x_i - x_{\text{med}}}{x_{75} - x_{25}}$$

Feature transformation

Some feature transformers:

- **Logarithmic** transformation: applies log function to features, which reduces skewness and non-linearity.



Feature transformation

Some feature transformers:

- **Logarithmic** transformation: applies log function to features, which reduces skewness and non-linearity
- **Power transform**: rescales features using power transformations, such as logarithm or square root, which makes them more Gaussian-like
- **Reciprocal** transformation: applies reciprocal function to features, which reduces skewness and non-linearity
- **Unit vector scaler**: rescales features by dividing by their magnitude, which makes them have a unit norm (good for word embeddings)

Be careful with feature scaling!

- Scaling can be a common source of **data leakage**
- Scaling variables requires global statistics (issue when dist shift)

4. Feature Selection

Why feature selection?

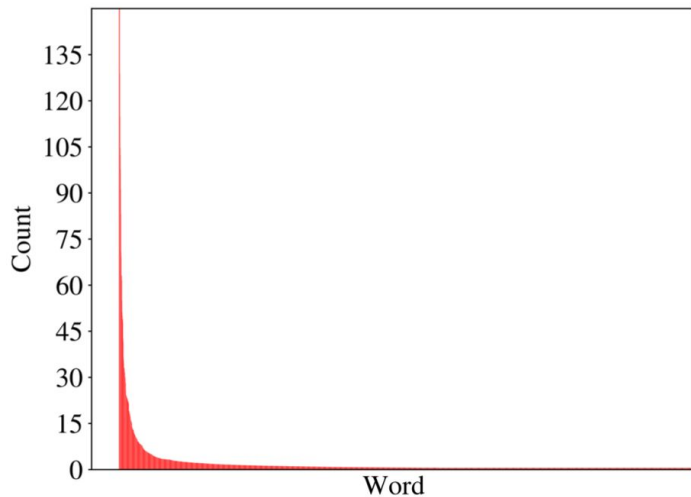
We use feature selection since:

- Improve **accuracy**: not all features are important for your problem (large dict in bag-of-words)
- Reduce **complexity**: the training time can become prohibitively long with large no. of features
- Avoid **overfitting**
- Reduce **technical debts** due to useless features.
- Improves **interpretability** of the results.

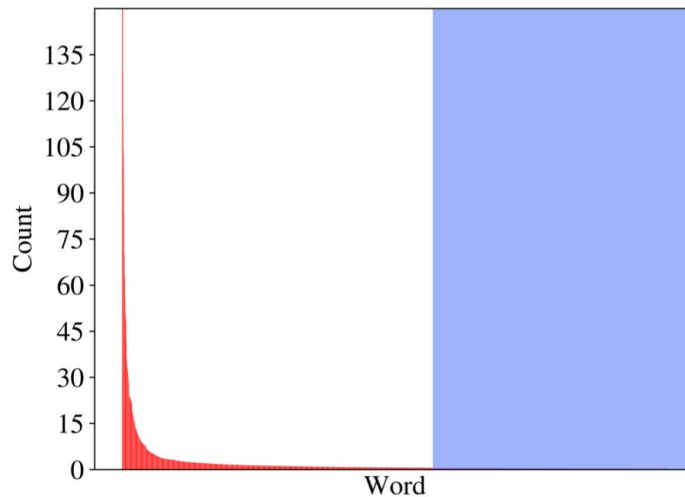
If we could **estimate** the **importance of features**, we would **select** only the most important ones.

Cutting the Long Tail

Removing long-tail features (rare tokens in bag-of-words) often results in faster learning and a better model.



(a) distribution of word counts in English



(b) the long tail

L1-Regularization

- Regularization is a term for techniques that improve the generalization of the model.
- Since L1 enforces sparsity, it implicitly performs feature selection by deciding which features are essential for prediction.

Boruta (spirit of the forests!)

A popular tool used in Kaggle competitions.

	age	height	weight		income
0	25	182	75	0	20
1	32	176	71	1	32
2	47	174	78	2	45
3	51	168	72	3	55
4	62	181	86	4	61



	age	height	weight	shadow_age	shadow_height	shadow_weight
0	25	182	75	51	176	75
1	32	176	71	32	182	71
2	47	174	78	47	168	78
3	51	168	72	25	181	72
4	62	181	86	62	174	86

are is useful only if it's capable of doing better than the best

Boruta (spirit of the forests!)

```
from sklearn.ensemble import RandomForestRegressor

### fit a random forest (suggested max_depth between 3 and 7)
forest = RandomForestRegressor(max_depth = 5, random_state = 42)
forest.fit(X_boruta, y)

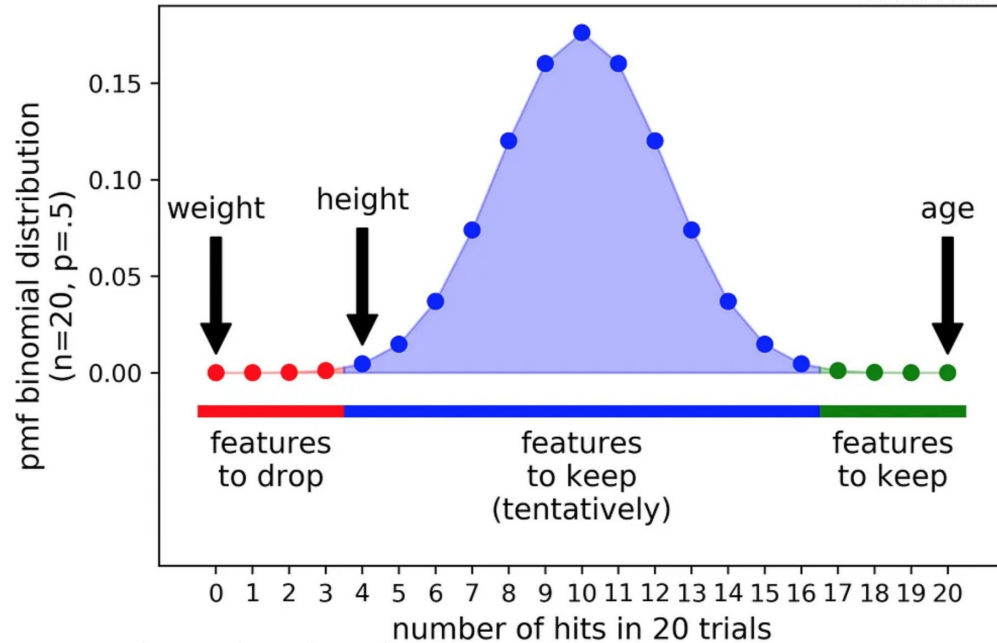
### store feature importances
feat_imp_X = forest.feature_importances_[0:len(X.columns)]
feat_imp_shadow = forest.feature_importances_[len(X.columns):]

### compute hits
hits = feat_imp_X > feat_imp_shadow.max()
```

	age	height	weight	shadow_age	shadow_height	shadow_weight
feature importance %	39	19	8	11	14	9
hits	1	1	0	-	-	-

	age	height	weight
hits (in 20 trials)	20	4	0

Boruta (spirit of the forests!)

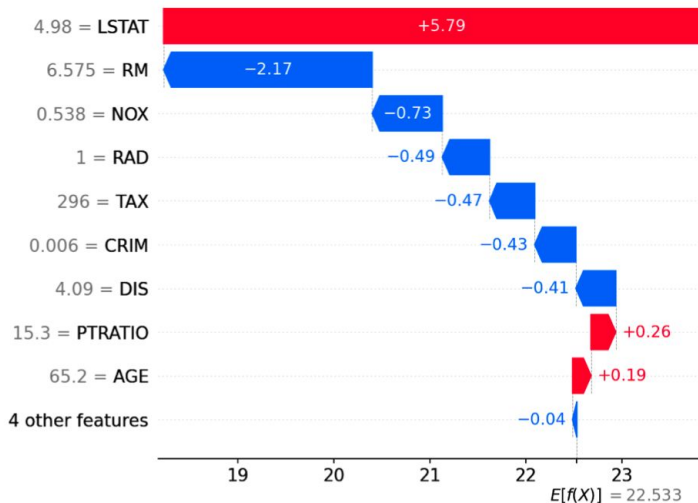


Feature importance

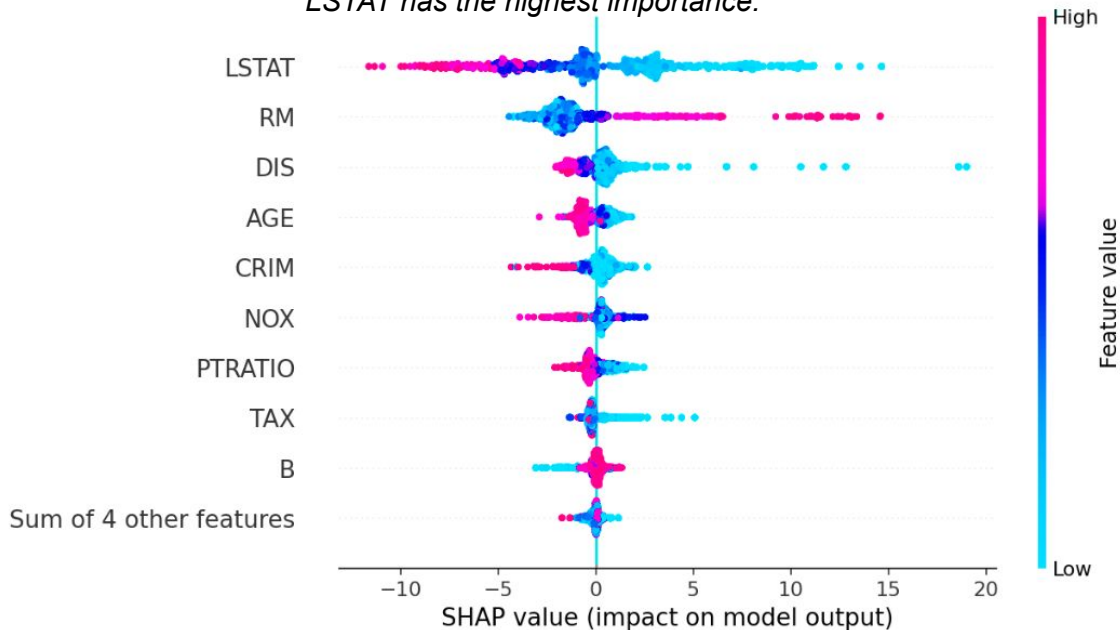
- Built-in feature importance functions in random forests and XGBoost.
- Model-agnostic methods, like SHAP (SHapley Additive exPlanations), which can help you understand the contribution of each feature to a model's predictions
- InterpretML library use feature importance for interpretability

Feature importance

The value LSTAT = 4.98 contributes the most to this specific prediction, measured by SHAP

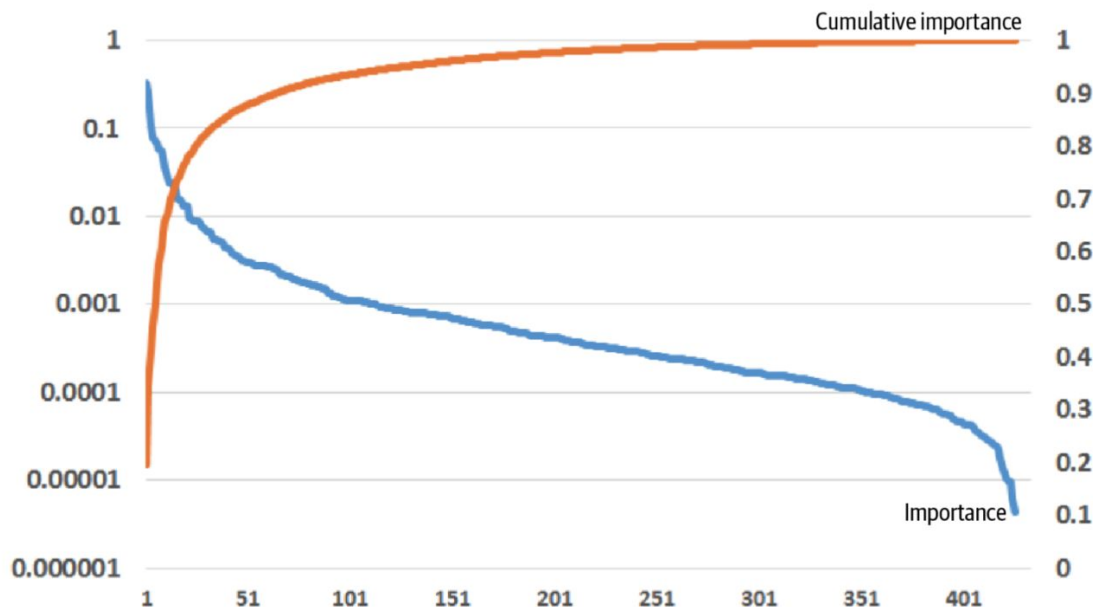


How much each feature contributes to a model, measured by SHAP. The feature LSTAT has the highest importance.



Feature importance

Often, a small number of features accounts for a large portion of your model's feature importance.



Ads team at Facebook found out that the **top 10** features are responsible for about **half** of the model's total feature importance, whereas the **last 300** features contribute less than **1% feature importance**

Recursive feature elimination

Suppose you have a data set with 10 features and you want to select the best 5 features for your model:

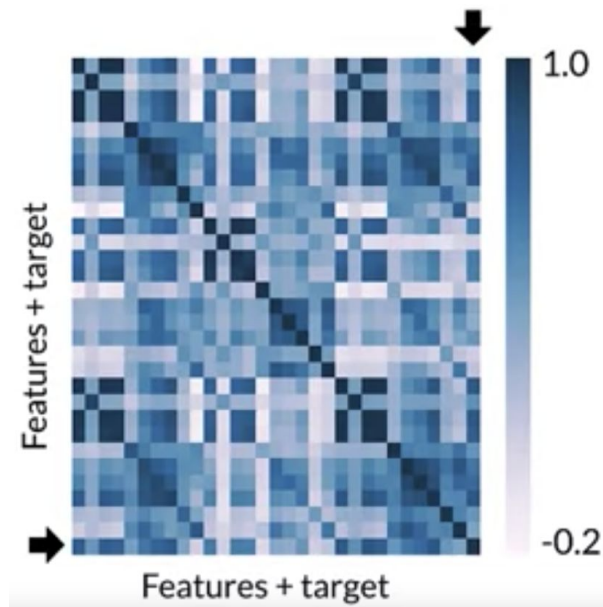
1. Fit a model using all 10 features and assign a score to each feature based on its importance or coefficient.
2. Remove the feature with the lowest score and fit a new model using the remaining 9 features.
3. Repeat step 2 until you have 5 features left.

You can use different models and scoring criteria for RFE, such as linear regression, logistic regression, xgboost.

Feature correlation

- Features that are highly **correlated** with the **target** variable are good predictors and should be **selected**.
- Features that are highly **correlated** with **each other** are redundant and should be **removed**.

Method	Feature Count	Accuracy	AUROC	Precision	Recall	F1 Score
All Features	30	0.967262	0.964912	0.931818	0.97619	0.953488
Correlation	21	0.974206	0.973684	0.953488	0.97619	0.964706



Feature generalization

Two aspects you might want to consider:

- **Feature coverage:** feature that appears in a very small percentage of your data, it's not going to be very generalizable
- **Distribution of feature values:** if the coverage of a feature differs a lot between the train and test split, probably test dist \neq train dist.

Summary: feature selection methods

- **Filter methods:** use statistical measures to score the correlation or dependence between input variables and select the most relevant features (e.g., correlation coefficient, chi-square test, ANOVA)
- **Wrapper methods:** use a machine learning algorithm to evaluate the performance of different subsets of features and select the best subset (e.g., forward selection, backward elimination, recursive feature elimination)
- **Embedded methods:** are part of the machine learning algorithms design and can automatically reduce the feature set during training (e.g., regularization techniques like Lasso, Ridge, Elastic Net)

Wrapper methods

- **Forward selection:** starts with an empty set of features and adds one feature at a time that improves the model performance (on dev set).
- **Backward elimination:** starts with all features and removes one feature at a time that has the least impact on the model performance (on dev set).
- **Recursive feature elimination**

Wrapper methods

Method	Direction	Criterion	Advantages	Disadvantages
Forward selection	Forward	Lowest p-value or highest accuracy	Simple, fast, suitable for small datasets	May miss optimal subset, prone to overfitting
Backward elimination	Backward	Highest p-value or lowest accuracy	Can find optimal subset, suitable for large datasets	Slow, complex, may overfit
Bi-directional elimination	Both	Lowest p-value or highest accuracy and highest p-value or lowest accuracy	Can balance between forward and backward methods, can find optimal subset	Very slow, very complex, may overfit

Filter methods

- **Correlation coefficient:** Removes features that are highly correlated with each other.
- **Variance threshold:** Removes features that have low variance.
- **Mutual information:** Selects features that have high mutual information with the target variable.

Embedded Methods

- **Regularization methods:** introduce additional constraints into the optimization of a predictive algorithm that bias the model toward lower complexity (fewer coefficients) (e.g. LASSO and RIDGE regression).
- **Tree-based methods:** use decision trees or ensembles of trees to rank features by their importance based on how they split the data (e.g., Decision Tree, RandomForest, ExtraTree, XGBoost)
- **Feature Importance:** is a technique to assign scores to input features based on how useful they are at predicting a target variable.

Machine Learning Systems Design

Data Lifecycle

Next Lecture: Feature Engineering (cont.)



CE 40959 Spring 2023

Ali Zarezade

[SharifMLSD.github.io](https://github.com/SharifMLSD)

TEMP slides

How to engineer good features

Evaluating a feature

1. Feature importance
2. Feature generalization

Measuring a feature's importance

How much the model performance deteriorates if a feature or a set of features containing that feature is removed from the model?

Measuring a feature's importance

- XGBoost

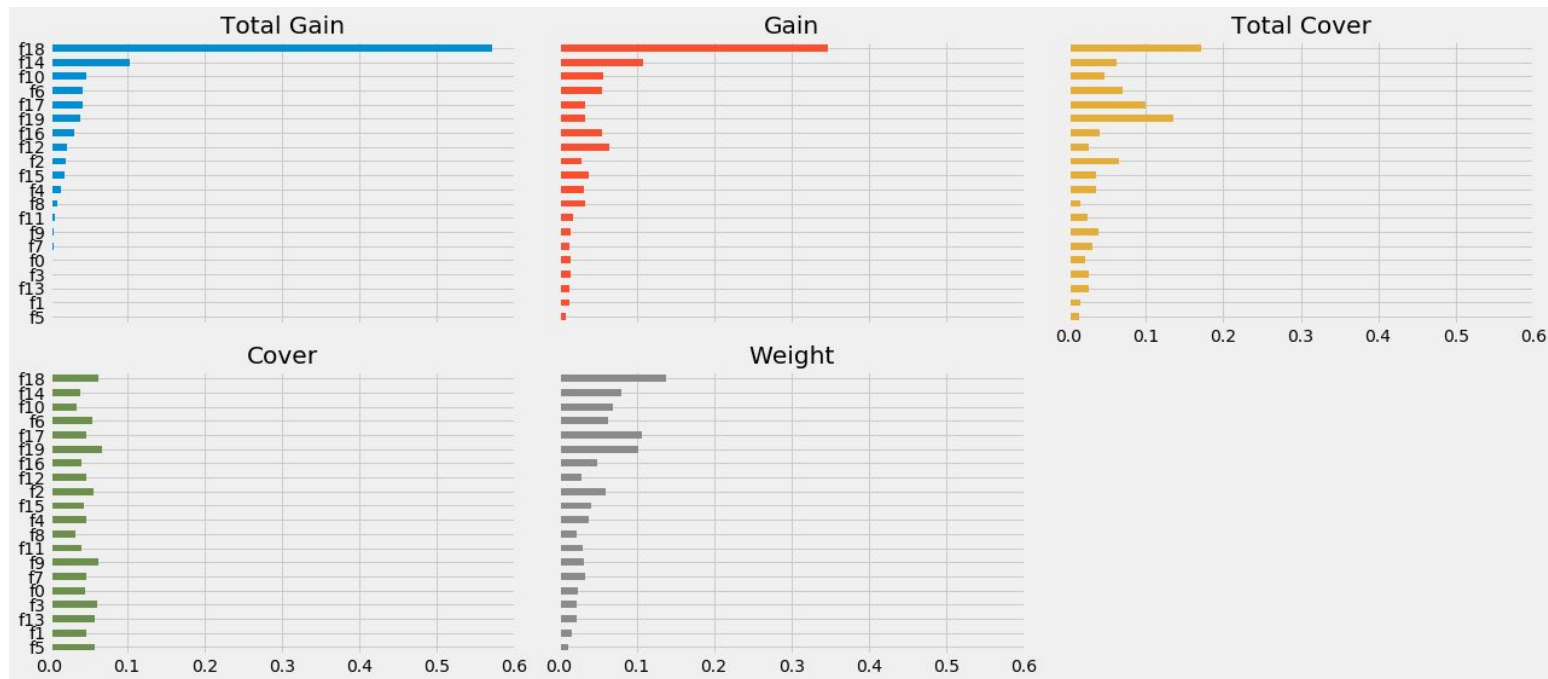
```
get_score(fmap="", importance_type='weight')
```

Get feature importance of each feature. For tree model Importance type can be defined as:

- 'weight': the number of times a feature is used to split the data across all trees.
- 'gain': the average gain across all splits the feature is used in.
- 'cover': the average coverage across all splits the feature is used in.
- 'total_gain': the total gain across all splits the feature is used in.
- 'total_cover': the total coverage across all splits the feature is used in.

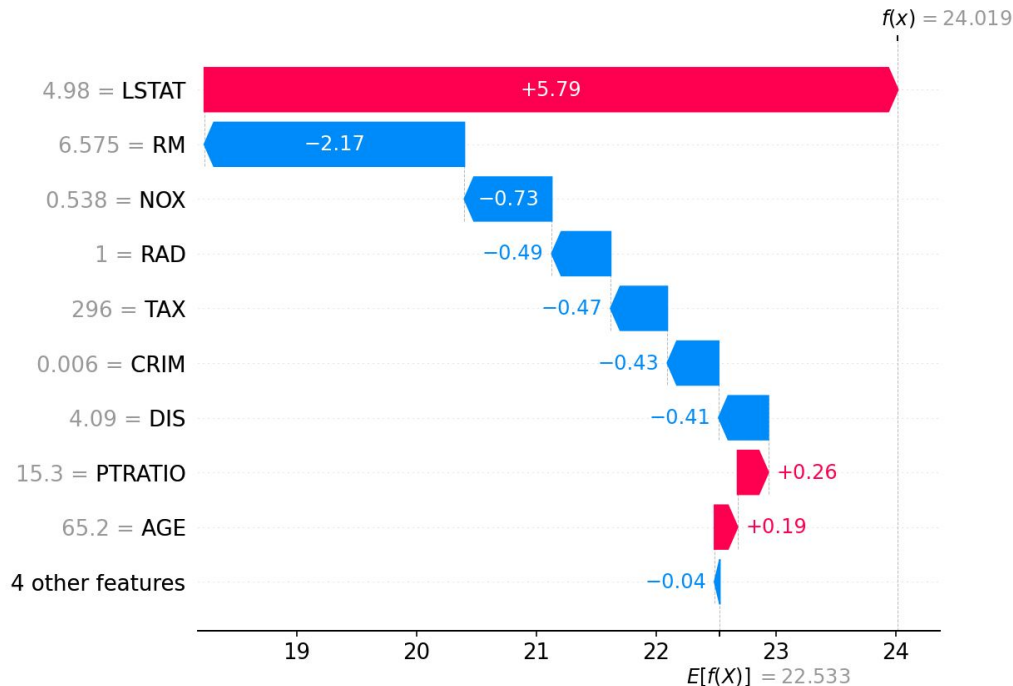
Measuring a feature's importance

- XGBoost



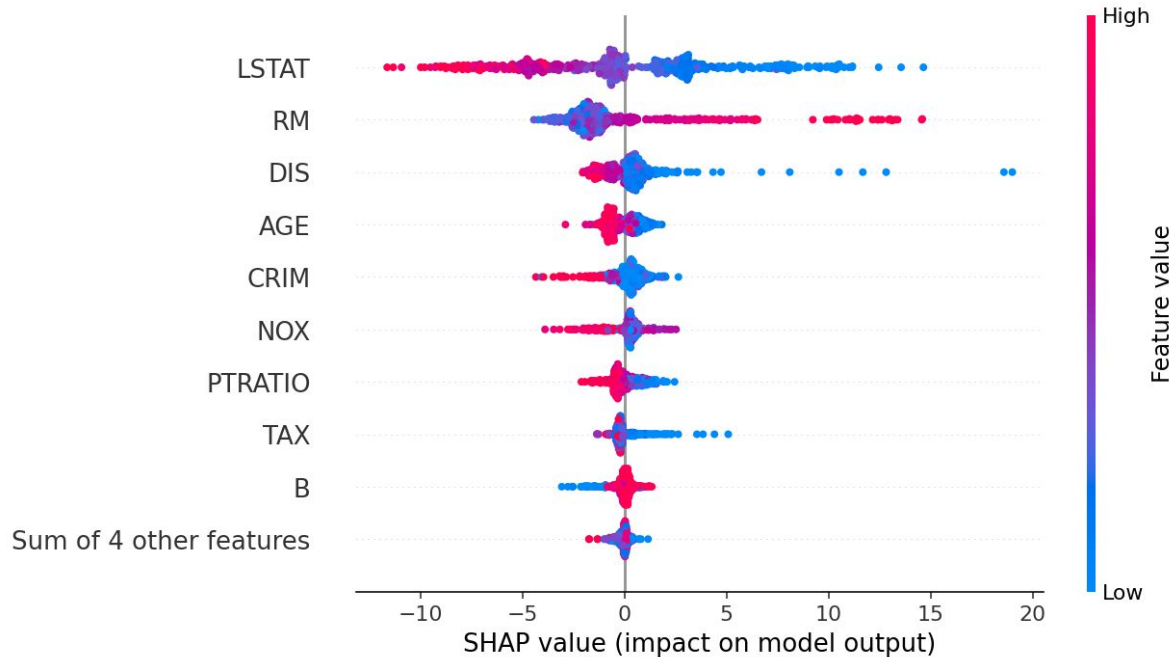
SHAP: SHapley Additive exPlanations

- Measuring a feature's contribution to a single prediction



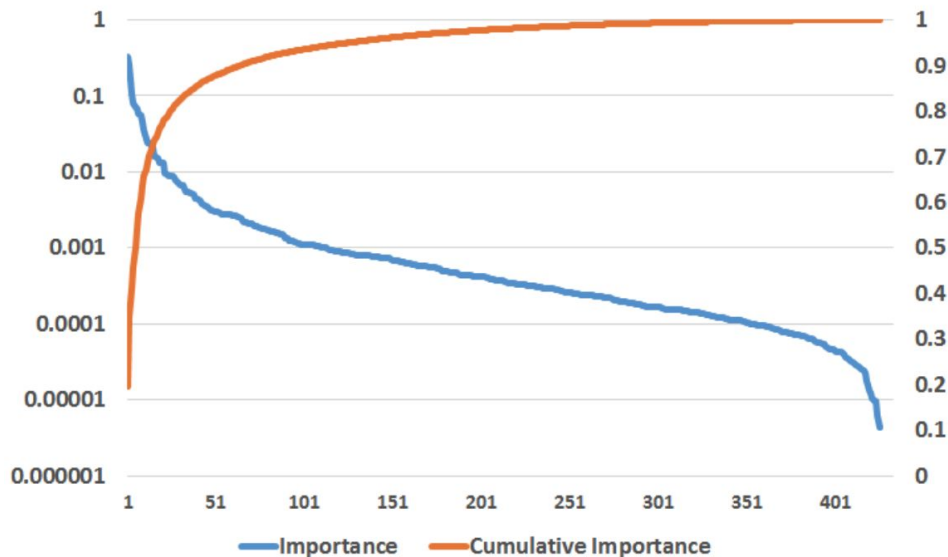
SHAP: SHapley Additive exPlanations

- Measuring a feature's contribution to **the entire model**



Measuring feature importance @ Facebook

- Top 10 features: 50% total feature importance
- Bottom 300 features: <1% total feature importance



```
ebm = ExplainableBoostingClassifier()  
I
```

Feature engineering: the more the better?

- Adding more features tends to improve model performance

How can having too many features be bad?

Too many features can be bad ...

- Training:
 - Overfitting
 - More features, more opportunity for data leakage

Too many features can be bad ...

- Training:
 - Overfitting
 - More features, more opportunity for data leakage
- Inference
 - Increase inference latency with online prediction
 - Might cause increased memory usage -> more expensive instance required

Too many features can be bad ...

- Training:
 - Overfitting
 - More features, more opportunity for data leakage
- Inference
 - Increase inference latency with online prediction
 - Might cause increased memory usage -> more expensive instance required
- Stale features become technical debts
 - E.g. if zip codes are no longer allowed for predictions, all features that use zip codes will need to be updated

Too many features can be bad ...

- Training:
 - Overfitting
 - More features, more opportunity for data leakage
- Inference
 - Increase inference latency with online prediction
 - Might cause increased memory usage -> more expensive instance required
- Stale features become technical debts

Solution:

- Clean up stale / ineffective features
- Store features in case you want to reuse them
 - Feature management

9 best practices for feature engineering

1. Split data by time instead of doing it randomly.
2. If you oversample your data, do it after splitting.
3. Use statistics/info from the train split, instead of the entire data, for feature engineering: scaling, normalizing, handling missing values, creating n-gram count, item encoding, etc.
4. Understand how your data is generated, collected, and processed. Involve domain experts if necessary.
5. Keep track of data lineage.
6. Understand feature importance to your model.
7. Measure correlation between features and labels.
8. Use features that generalize well.
9. Remove stale features from your models.