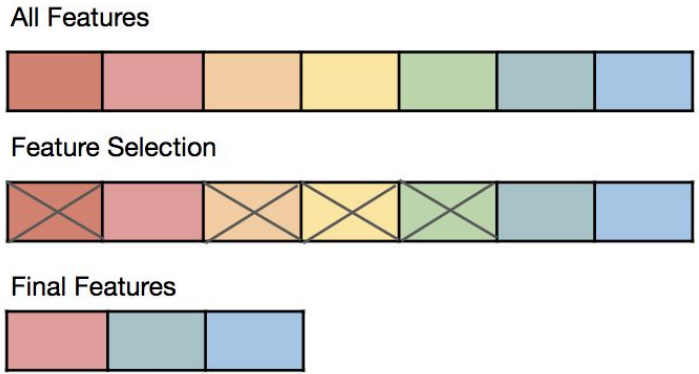
Feature Importance for Black Box Models

By Will Holt and Michael Wise
DATA 477 Capstone

What is Feature Importance?

- Variable selection is essential in building robust ML models.
- Not all input data features are equally important for making accurate and efficient models.
- Reducing the number of features can reduce computational cost and improve model performance.
- **Feature importance** - any technique that calculates a score or weight for all of the input variables given some prediction model.
- The goal of feature importance is to find features with the highest or lowest scores/weights, depending on the context.
- Remove predictors that are unimportant to our model



P-Value Importance

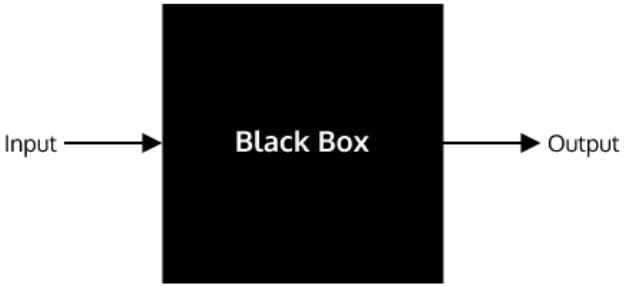
Coefficients

Term	Coef	SE Coef	T	P
Constant	389.166	66.0937	5.8881	0.000
East	2.125	1.2145	1.7495	0.092
South	5.318	0.9629	5.5232	0.000
North	-24.132	1.8685	-12.9153	0.000

- p-value: a measure that the correlation of the output variable to the input variable could occur by chance.
- By observing the probability of observing a t-statistic (difference in actual coeff. and hypothesized coeff. divided by standard error) we can derive p-value
- If the p-value is less than a pre-specified significance level (typically 0.05), we reject the null hypothesis and conclude that the coefficient is statistically significant.
- **Only works for regression where we clearly see the impact of coefficients/predictors**

The Black Box Problem

- Feature importance isn't the same for every model on a data set
  - Some predictors work better on certain models, and not others
- I.e. neural networks only produce a series of weighted nodes for variables to travel through
  - There is no way to determine the effectiveness of an individual feature from the weights and model alone.
- "Black box problem" - lack of transparency and interpretability of ML models
  - Ability to make accurate predictions, but model's internal workings of the model are not easily understandable or interpretable.
- How can we figure out what features are the most important in our model?



Examples:

- Neural networks
- SVMs
- Random forests/trees
- Gradient boosting models

Cannot extract the impact of the feature just from the model itself like we can with regression.

Permutation Importance

- Possible solution: permutation feature importance (PFI)
- Taking each variable one at a time and permuting all of the values in that variable.
- Re-train the model and measure the difference in accuracy

Logic:

- If the model's accuracy is very similar after permutation -> feature (likely) **not** very important and can be removed.
- If the accuracy drops significantly -> feature (likely) very important and shouldn't be removed.

PFI Importance Algorithm

Given fitted predictive model m, and tabular dataset D:

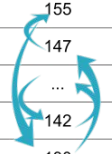
- Compute the reference score of the model m on data D (most commonly accuracy for classification or R^2 for regression)
- For each feature j (column of D):
  - For each repetition k in 1, … , K:
    - Randomly shuffle column j of D to generate a randomized copy of data called $\widehat{D}_{k,j}$.
    - Compute score s_{kj} of model m on corrupted data
  - Compute importance i_j for feature f_j defined as:

$$i_j = s - \frac{1}{K}\sum_{k=1}^{K} s_{k,j}$$

Issues with PFI

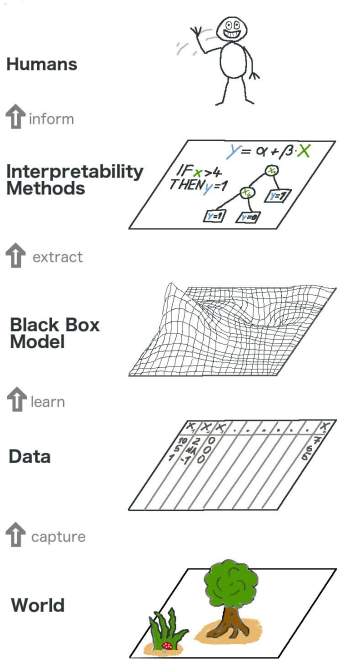
- Highly correlated values will show up as far less important than they actually are because the information of the one feature is not actually lost during permutation
- Not an accurate method unless there is very little variable correlations
- Depends heavily on the performance of the model
    - Features that could be found with low importance for a bad model could be very important for a well-performing model.
- However, since it **only** relies on the data and the accuracy score, we can use it on **any model** (model-agnostic)

Height at age 20 (cm)	Height at age 10 (cm)	...	Socks owned at age 10
182	155	...	20
175	147	...	10
...	...	...	...
156	142	...	8
153	130	...	24



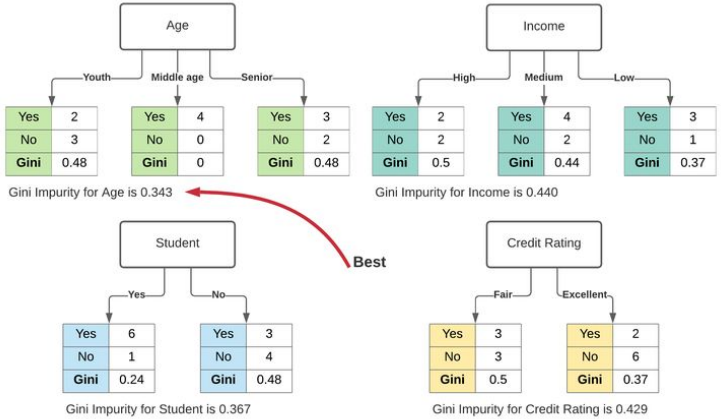
Drop-and-Relearn Importance

- “Leave-one-covariate-out” method
- Very similar to PFI, but instead of permuting the column of a variable, we **remove it entirely.**
- Re-train and measure change in accuracy
- Based off of this, decision is then made as to whether or not the variable should be removed.
- Same issue with PFI: doesn’t work well with data sets with many correlated features
- Again, it is model-agnostic



Mean Decrease in Impurity/Gini Importance

- Tree/ensemble models only!
- Total decrease in node impurity
  - weighted by the probability of reaching that node (which is approximated by the proportion of samples reaching that node) averaged over all trees of the ensemble
- Using this terminology, we could call the PFI/Drop-and-Relearn techniques that measure “mean decrease in accuracy”
- Were just comparing what the largest decrease is to know which variables have the most impact



Always take feature split with the lowest weighted Gini impurity - this is how many trees are trained

LIME

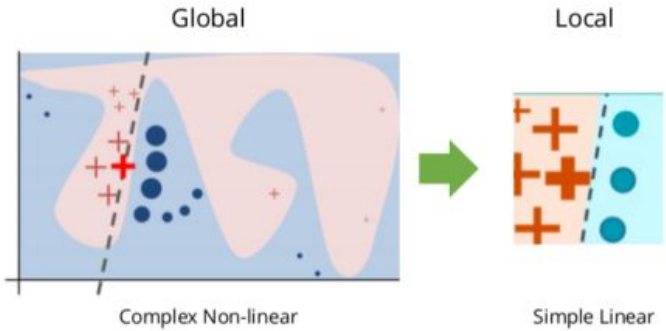
- Local Interpretable Model-Agnostic Explanations (LIME)
- Perturb the input data around the instance of interest (usually a single observation in the data) and measure the effect of feature on the model's prediction
- Add noise to original data, feed it back into model, and assign weights to the new data points as some function of their proximity to the original point.
- Result: set of weighted features that we then train on a **"local surrogate model"**
- We can now represent any black box model and treat it like a linear regression

$$\text{explanation}(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g)$$

- x = instance
- f = original (black box) model
- g is the surrogate model that minimizes loss L (usually mean squared or absolute error)
- Ω(g) -model complexity (try to minimize)
- π_x = the proximity measure, which defines how large the neighborhood around our instance that we consider for our explanation.

LIME Analysis

- Super model-agnostic
  - Even if you replace the underlying ML model, you can continue to use the same local, interpretable model
- Human friendly explanations
- Able to support more than just tabular data (text & image as well)
  - All that changes is that pixels are our features!!
- Categorical features (especially for trees/boosting methods) can be interpreted well
- The proximity measure π doesn't work as well as our data becomes more high dimensional
- Method is still in development phase as explanations have been shown to be unstable



Shapley Values

- We can use coalitional game theory to calculate feature importance
- The **Shapley value** of a feature value (predictor) is its contribution to the payout (prediction), weighted and summed over all possible feature value combinations:

$$\phi_j(val) = \sum_{S \subseteq \{1,\ldots,p\} \setminus \{j\}} \frac{|S|!\,(p - |S| - 1)!}{p!} (val\,(S \cup \{j\}) - val(S))$$

$val_x(S)$ = predictor for feature values in set S that are marginalized over features not included in set S

$$val_x(S) = \int \hat{f}\,(x_1, \ldots, x_p) d\mathbb{P}_{x \notin S} - E_X(\hat{f}\,(X))$$

While providing accurate feedback on feature importance, most Shapley values are impossible to derive exactly/analytically as we add more feature values to our data.

We would have to evaluate all possible coalitions (sets) of feature values with and without the *j*-th feature - computational nightmare.

Approximating Shapley Values

We can approximate Shapley values with Monte-Carlo sampling:

$$\hat{\phi}_j = \frac{1}{M} \sum_{m=1}^{M} \left(\hat{f}\left(x_{+j}^m\right) - \hat{f}\left(x_{-j}^m\right) \right)$$

Take original observations for features that appear **left** of x_j

Take random instance observations for features that appear **right** of x_j

$\hat{f}\left(x_{+j}^m\right)$ is the prediction for x, except we replace a random number of feature values from *z* (random data point)

We create M of these new “hybrid” instances.

SHAP (SHapley Additive exPlanations)

- Goal of SHAP: explain the prediction of an instance by computing the contribution of each feature to the prediction
- Computes Shapley values from coalitional game theory
  - Tell us how to fairly distribute the “payout” (prediction) amount the “players” (features)

Explanation Model:

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j$$

Simplified Model assuming x’ (vector of all 1’s):

$$g(x') = \phi_0 + \sum_{j=1}^{M} \phi_j$$

$$z' \in \{0,1\}^M$$

Coalition Vector
1 = feature value is “present”
0 = feature value is “absent”

$$\phi_j \in \mathbb{R}$$

Feature Attribution for feature *j*

M = maximum feature size

SHAP Advantages/Disadvantages

$$I_j = \frac{1}{n} \sum_{i=1}^{n} |\phi_j^{(i)}|$$

Advantages

- Difference between prediction and average prediction is fairly distributed (LIME doesn't guarantee this)
- Allows for contrastive explanations
  - Can compare to avg. prediction of entire dataset, or even a subset or single data point
- Rooted in solid game theory with its own axioms and theorems
- Fast implementation for tree-based models and doesn't get impacted by feature dependence

Disadvantages

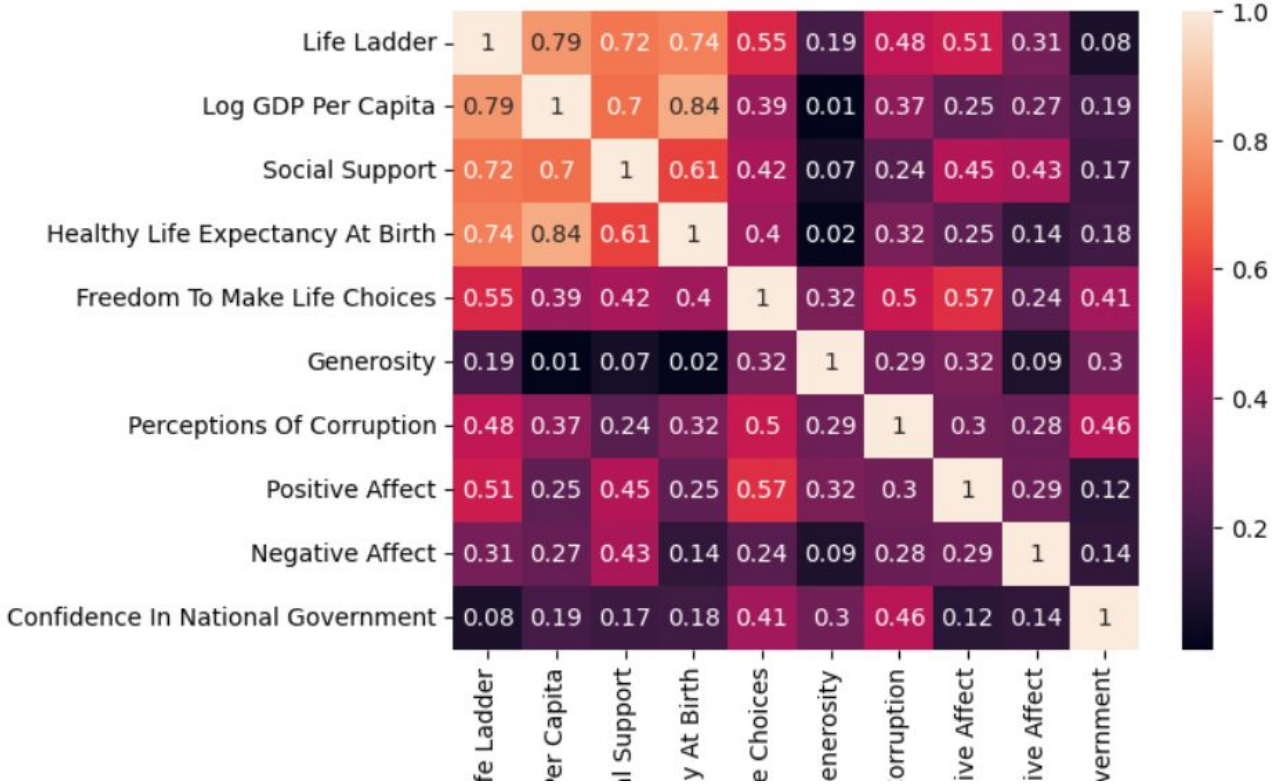
- Requires a lot of computational time/power
- Shapley values can be misinterpreted
  - Contribution a feature value to the difference between the actual prediction and the mean prediction is the estimated, **NOT** the diff. in predicted value after removing a feature from training
- Explanations always use all the predictors (not selective)

Example: World Happiness Report

Life Ladder	Log GDP Per Capita	Social Support	Healthy Life Expectancy At Birth	Freedom To Make Life Choices
3.723590	7.350416	0.450662	50.500000	0.718114
4.401778	7.508646	0.552308	50.799999	0.678896
4.758381	7.613900	0.539075	51.099998	0.600127
3.831719	7.581259	0.521104	51.400002	0.495901
3.782938	7.660506	0.520637	51.700001	0.530935

Generosity	Perceptions Of Corruption	Positive Affect	Negative Affect	Confidence In National Government
0.167652	0.881686	0.414297	0.258195	0.612072
0.190809	0.850035	0.481421	0.237092	0.611545
0.121316	0.706766	0.516907	0.275324	0.299357
0.163571	0.731109	0.479835	0.267175	0.307386
0.237588	0.775620	0.613513	0.267919	0.435440

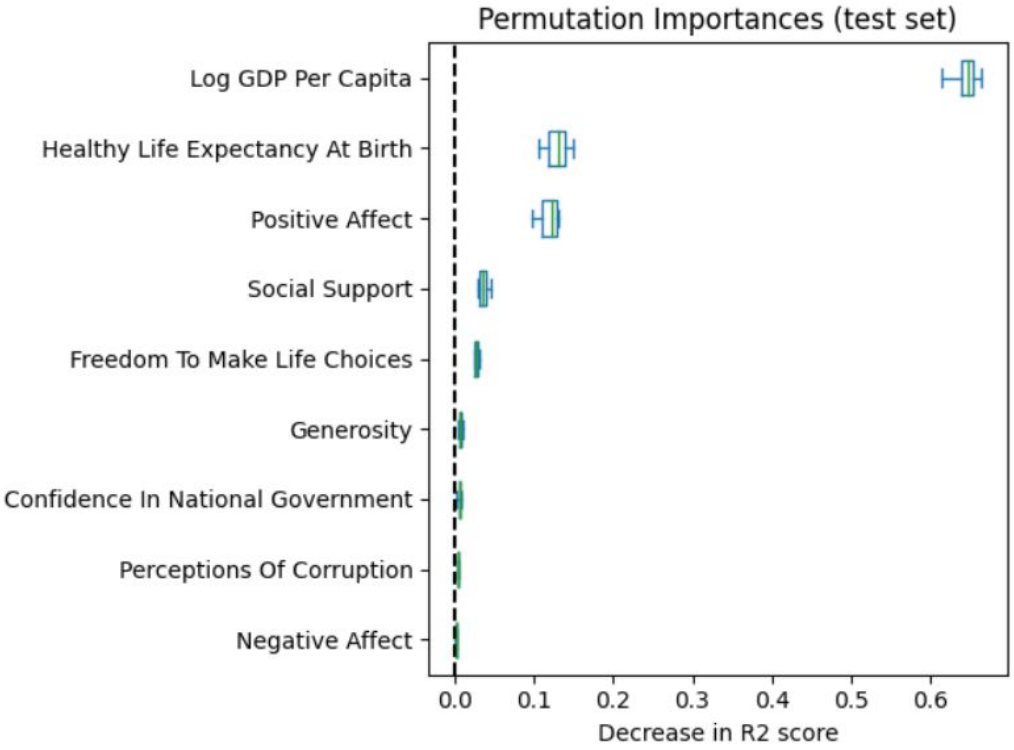
Correlation Matrix



Regression P-value

	Variables	p-value
1	Log.GDP.Per.Capita	2.695095e-49
7	Positive.Affect	3.798996e-34
6	Perceptions.Of.Corruption	1.514608e-30
2	Social.Support	8.032279e-21
9	Confidence.In.National.Government	6.081387e-14
3	Healthy.Life.Expectancy.At.Birth	1.358115e-13
0	(Intercept)	2.889320e-11
4	Freedom.To.Make.Life.Choices	1.456266e-08
5	Generosity	3.145402e-08
8	Negative.Affect	9.362452e-01

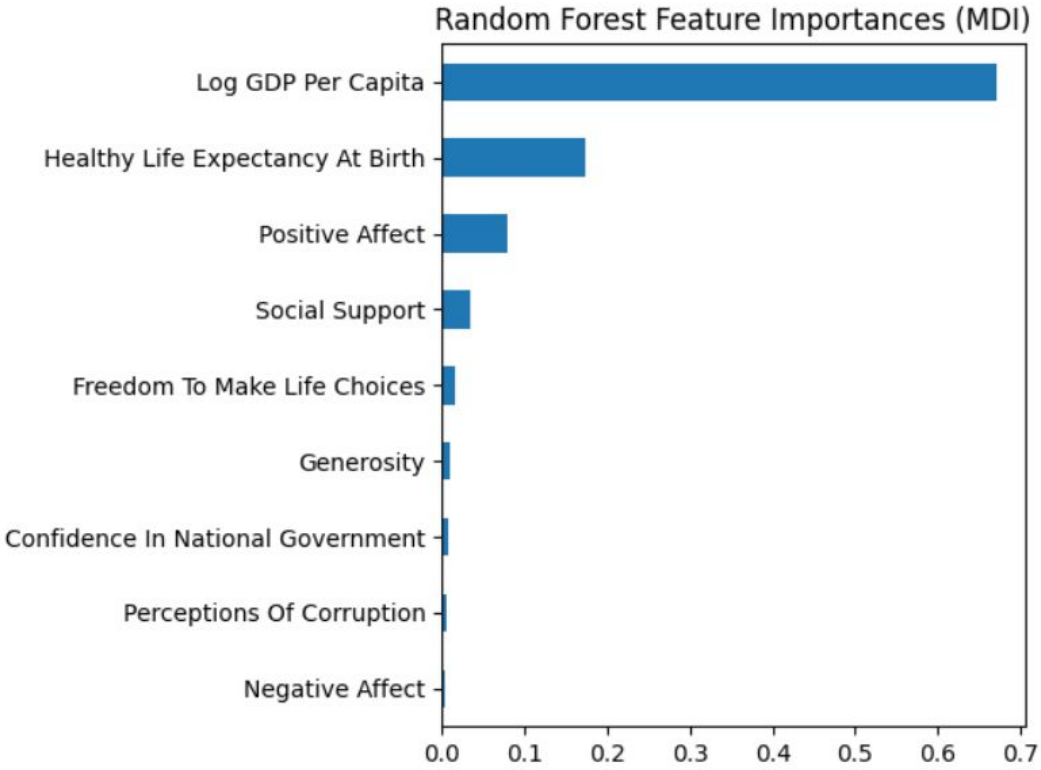
Permutation Importance Example



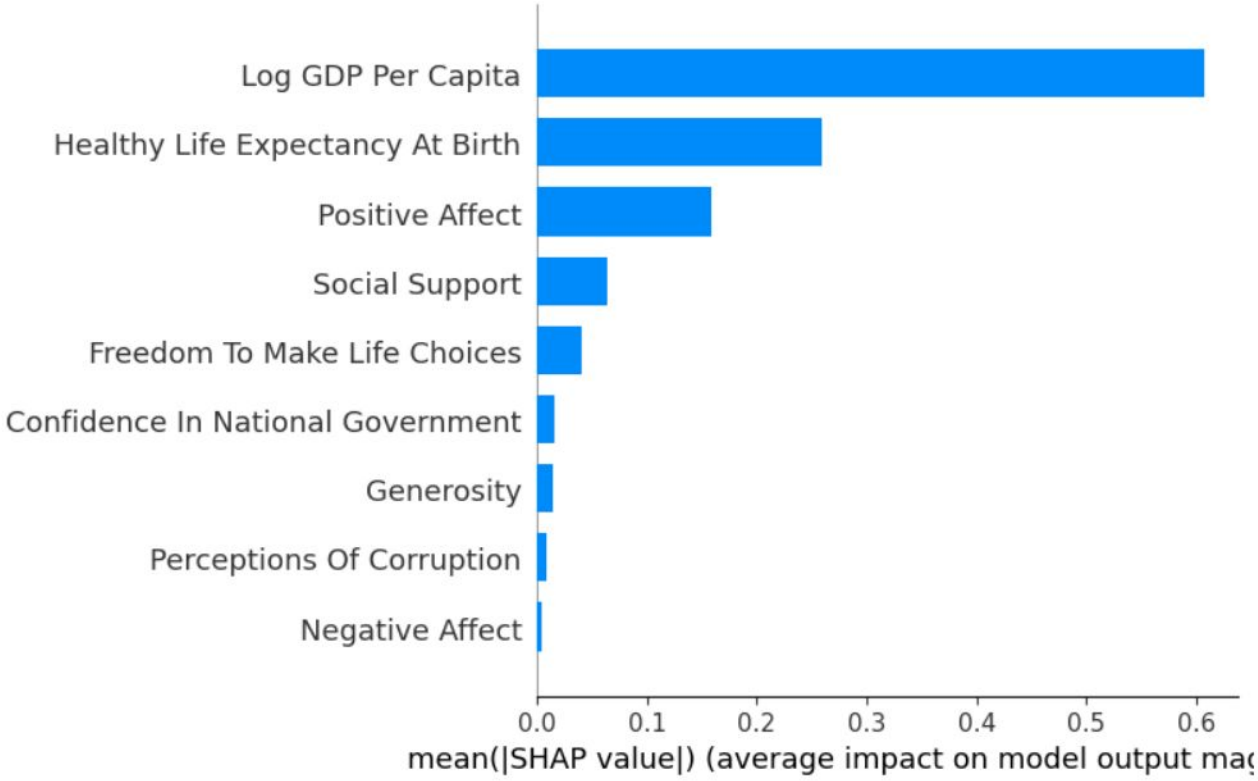
Drop-and-Relearn Example

	Variable Removed	R2	Difference
1	Log GDP Per Capita	0.760909	-0.024127
7	Positive Affect	0.762511	-0.022525
6	Perceptions Of Corruption	0.764622	-0.020414
2	Social Support	0.769697	-0.015340
9	Confidence In National Government	0.777264	-0.007772
5	Generosity	0.779451	-0.005586
4	Freedom To Make Life Choices	0.779688	-0.005349
3	Healthy Life Expectancy At Birth	0.784138	-0.000898
0	None	0.785037	0.000000
8	Negative Affect	0.785109	0.000072

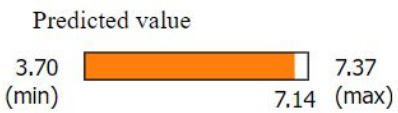
MDI Example


Random Forest Feature Importances (MDI)
Log GDP Per Capita
Healthy Life Expectancy At Birth
Positive Affect
Social Support
Freedom To Make Life Choices
Generosity
Confidence In National Government
Perceptions Of Corruption
Negative Affect
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7

Shap Example

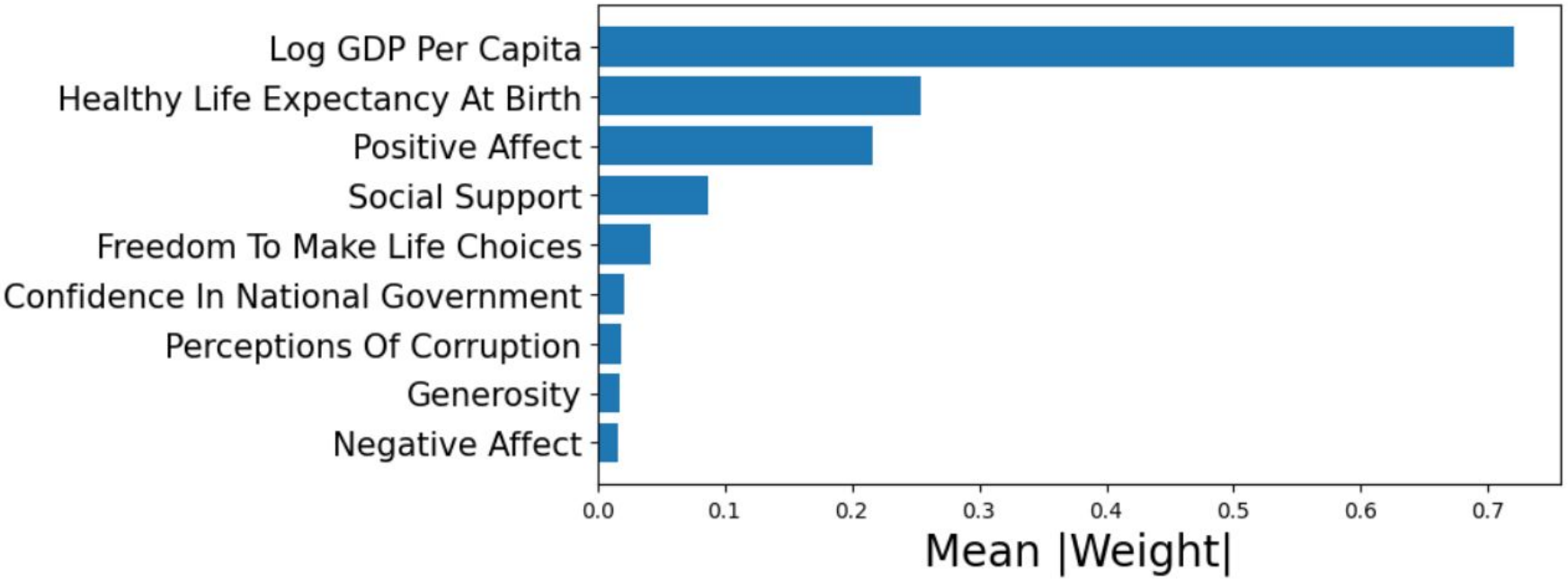


LIME Example



Feature	Value
Log GDP Per Capita	10.65
Healthy Life Expectancy At Birth	69.46
Social Support	0.95
Positive Affect	0.74
Freedom To Make Life Choices	0.90
Confidence In National Government	0.47
Negative Affect	0.17
Perceptions Of Corruption	0.44
Generosity	0.33

LIME Mean


Log GDP Per Capita
Healthy Life Expectancy At Birth
Positive Affect
Social Support
Freedom To Make Life Choices
Confidence In National Government
Perceptions Of Corruption
Generosity
Negative Affect
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Mean |Weight|

Final Comparison

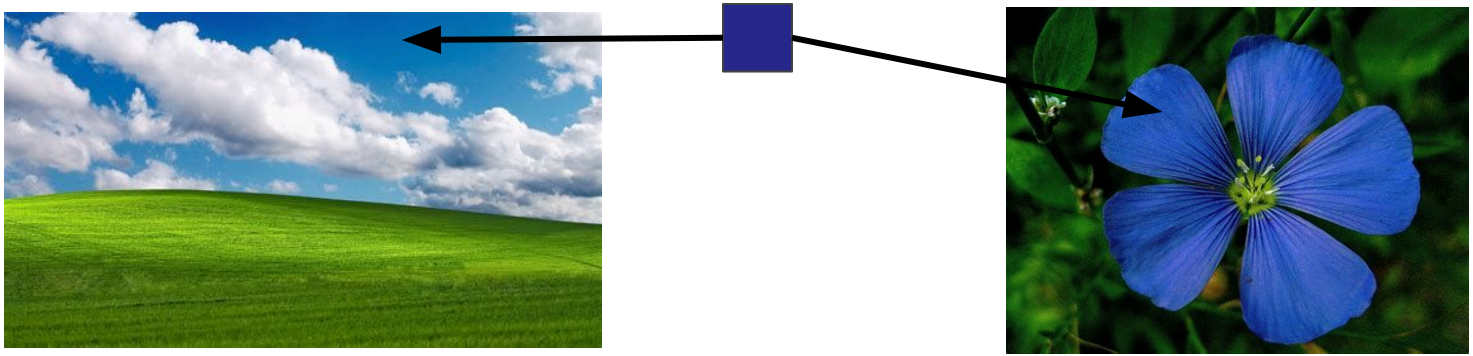
	P-Value	DnR LR	DnR RF	MDI	Permutation	SHAP	Lime
1	Log.GDP.Per.Capita	Log GDP Per Capita	Log GDP Per Capita	Log GDP Per Capita	Log GDP Per Capita	Log GDP Per Capita	Log GDP Per Capita
2	Social.Support	Social Support	Social Support	Healthy Life Expectancy At Birth	Healthy Life Expectancy At Birth	Healthy Life Expectancy At Birth	Healthy Life Expectancy At Birth
3	Healthy.Life.Expectancy.At.Birth	Healthy Life Expectancy At Birth	Healthy Life Expectancy At Birth	Positive Affect	Positive Affect	Positive Affect	Positive Affect
4	Freedom.To.Make.Life.Choices	Freedom To Make Life Choices	Freedom To Make Life Choices	Social Support	Social Support	Social Support	Social Support
5	Generosity	Generosity	Generosity	Freedom To Make Life Choices	Freedom To Make Life Choices	Freedom To Make Life Choices	Freedom To Make Life Choices
6	Perceptions.Of.Corruption	Perceptions Of Corruption	Perceptions Of Corruption	Generosity	Confidence In National Government	Confidence In National Government	Confidence In National Government
7	Positive.Affect	Positive Affect	Positive Affect	Confidence In National Government	Generosity	Generosity	Perceptions Of Corruption
8	Negative.Affect	Negative Affect	Negative Affect	Perceptions Of Corruption	Perceptions Of Corruption	Perceptions Of Corruption	Generosity
9	Confidence.In.National.Government	Confidence In National Government	Confidence In National Government	Negative Affect	Negative Affect	Negative Affect	Negative Affect

Key Takeaways

- Of the 2 basic methods, permutation importance seems to be the most accurate
- SHAP, LIME, and MDI require large amounts of computing power, but are highly accurate
- When resources are limited, permutation importance should be prioritized over drop-and-relearn
- There is no one perfect method

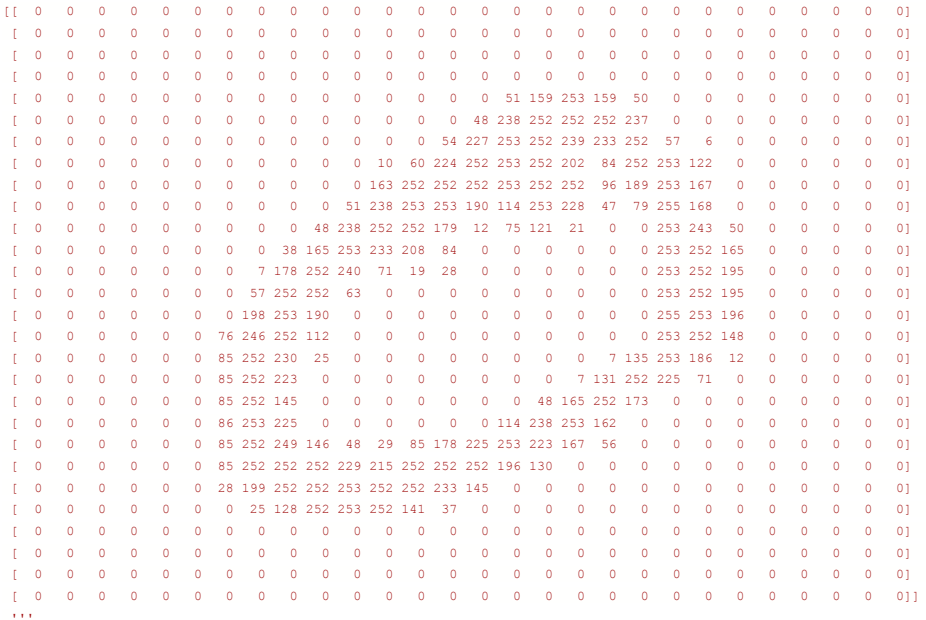
Image Feature Importance

- Unlike normal features, pixels have no innate meaning
- Groups of pixels or segments give context
- Feature importance can be done on these segments



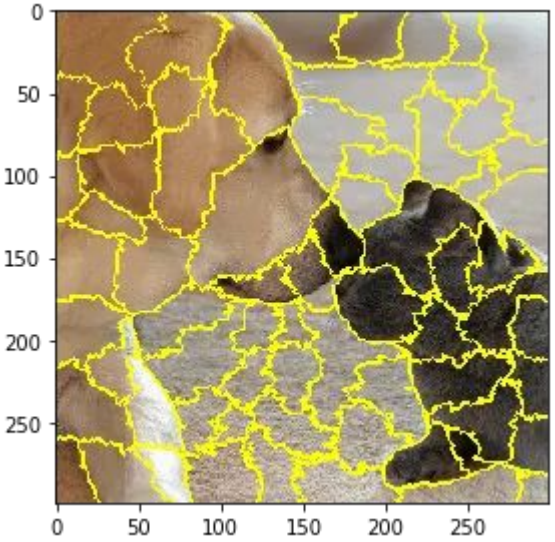
Pixel Segmentation

- Not very useful
- Gives a heatmap of pixels that change the most
- Dependent on object placement within the image
- Minor insights at best



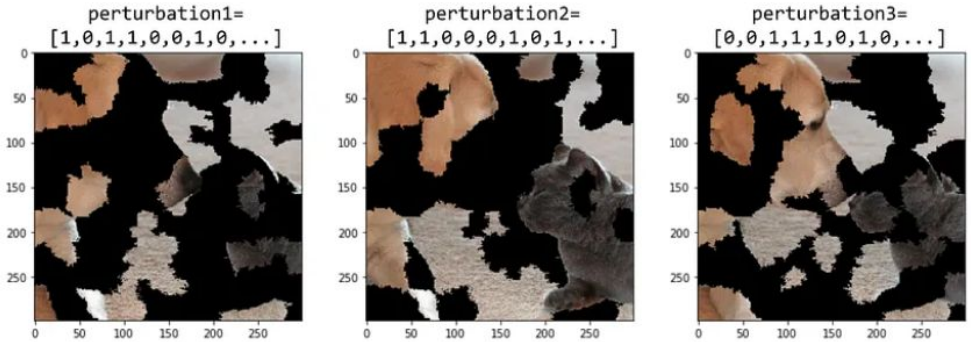
Segmentation

- Groups of pixels that share similar qualities
  - Color, intensity, texture
- Used to find objects and boundaries
- Useful for image classification



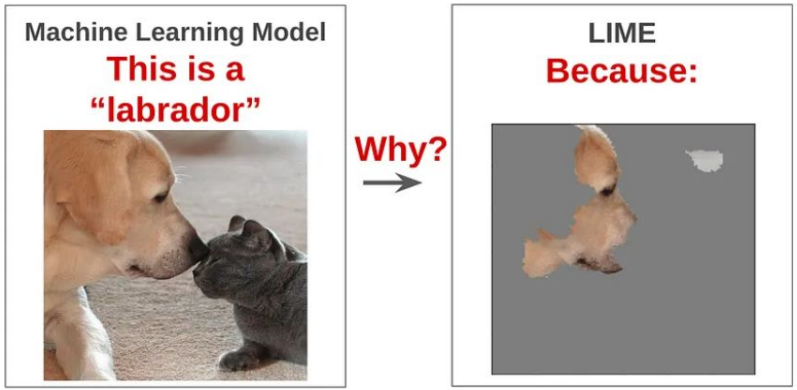
Segmentation Permutation

- Permutates the image's segments by randomly selecting which segments to test on
- Used in many image feature importance algorithms



LIME for Images

- Creates random segment permutations
- Scores each permutation based on model accuracy
- Compares results to determine most important segments

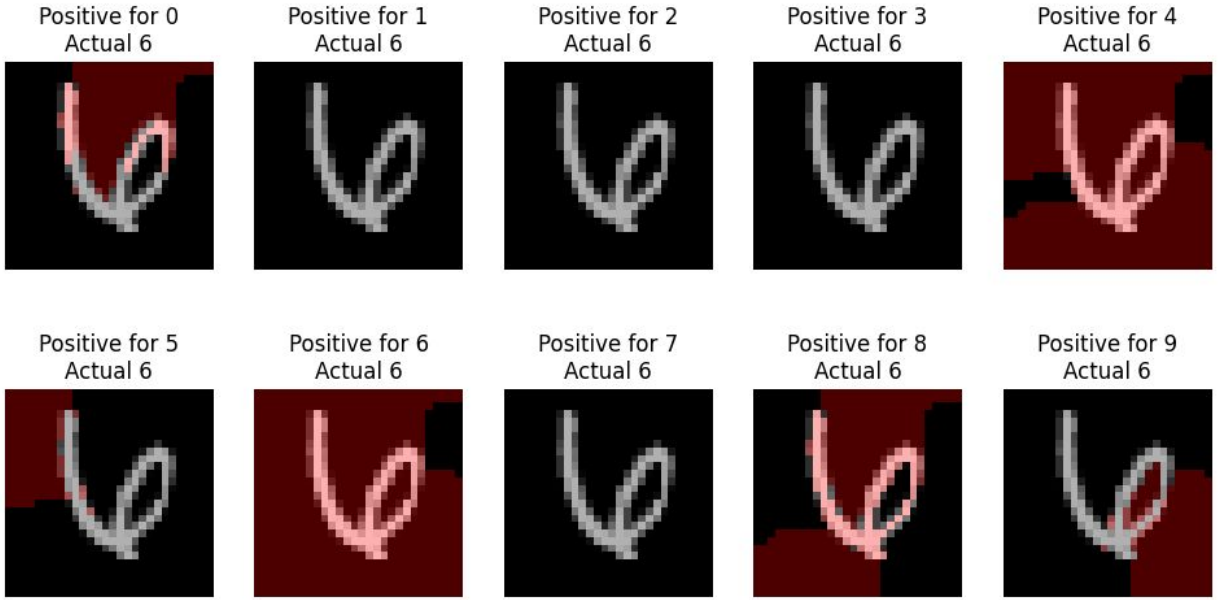


Example: MNIST Digits

- Handwritten numbers by hundreds of government employees in 1995
- 60,000 labeled numbers
- 28x28 pixels in size
- Relatively easy machine learning problem

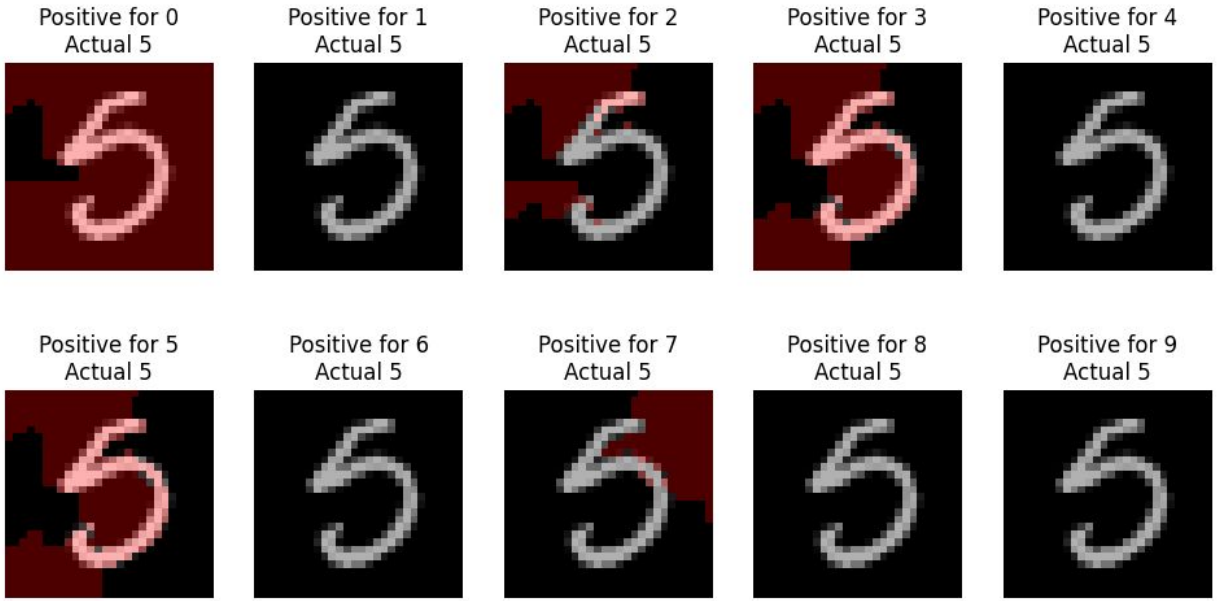
LIME Positive Example

Positive for 6, Actual 6



LIME Negative Example

Positive for 0, Actual 5



Future Work

- Further research on image feature importance
- Trying feature importance methods for more model types.
  - Neural Networks, XGboost, SVMs, etc.
- Using feature importance methods to shrink models

References

https://towardsdatascience.com/interpretable-machine-learning-for-image-classification-with-lime-ea947e82ca13

https://github.com/marcotcr/lime/blob/master/doc/notebooks/Tutorial%20-%20MNIST%20and%20RF.ipynb

https://www.gallup.com/analytics/349487/gallup-global-happiness-center.aspx

https://www.mathworks.com/discovery/image-segmentation.html

https://towardsdatascience.com/how-to-find-feature-importances-for-blackbox-models-c418b694659d

https://towardsdatascience.com/interpret-your-black-box-ml-model-with-permutation-feature-importance-fc2b2a14ca7c

http://yann.lecun.com/exdb/mnist/