Interpretability

ACTL3143 & ACTL5111 Deep Learning for Actuaries

Patrick Laub

Lecture Outline

- **Interpretability**
- Inherent Interpretability
- Post-hoc Interpretability
- Illustrative Example
- Illustrative Example (Fixed)

Interpretability

Interpretability Definition

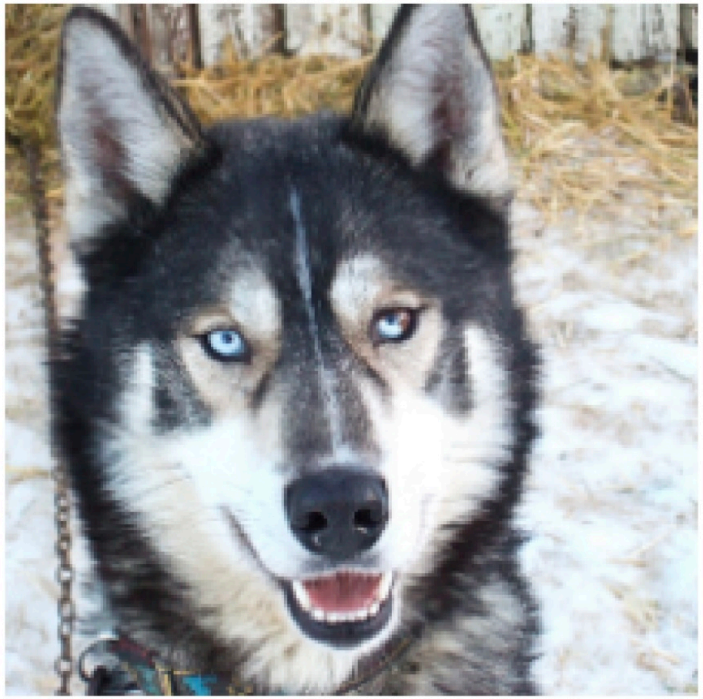
Interpretability refers to the ease with which one can understand and comprehend the model's algorithm and predictions.

Interpretability of black-box models can be crucial to ascertaining trust.

> Interpretability is about transparency, about understanding exactly why and how the model is generating predictions, and therefore, it is important to observe the inner mechanics of the algorithm considered. This leads to interpreting the model's parameters and features used to determine the given output. Explainability is about explaining the behavior of the model in human terms.

Source: Charpentier (2024), *Insurance, Biases, Discrimination and Fairness*, Springer.

Husky vs. Wolf



(a) Husky classified as wolf

(b) Explanation

Figure 11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.

A well-known anecdote in the explainability literature.

Ribeiro et al. (2016), *"Why should I trust you?" Explaining the predictions of any classifier*, 22nd ACM SIGKDD conference.

Aspects of Interpretability

Inherent Interpretability

The model is interpretable by design.

Post-hoc Interpretability

The model is not interpretable by design, but we can use other methods to explain the model.

Global Interpretability

The ability to understand how the model works.

Local Interpretability

The ability to interpret/understand each prediction.

Lecture Outline

- Interpretability
- **Inherent Interpretability**
- Post-hoc Interpretability
- Illustrative Example
- Illustrative Example (Fixed)

Perspective | Published: 13 May 2019

Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead

Cynthia Rudin

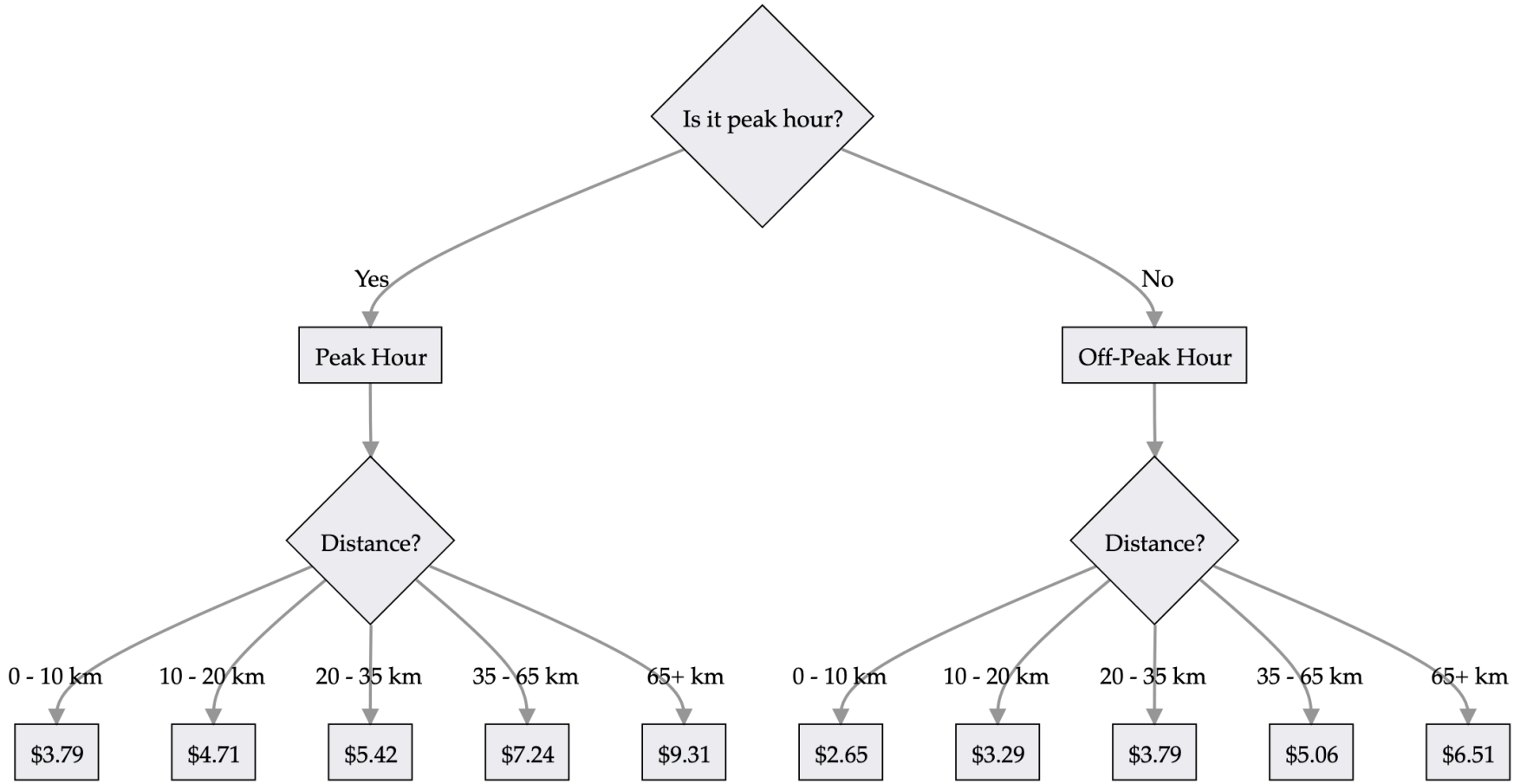
Nature Machine Intelligence **1**, 206–215 (2019) | Cite this article

66k Accesses | **2230** Citations | **485** Altmetric | Metrics

Abstract

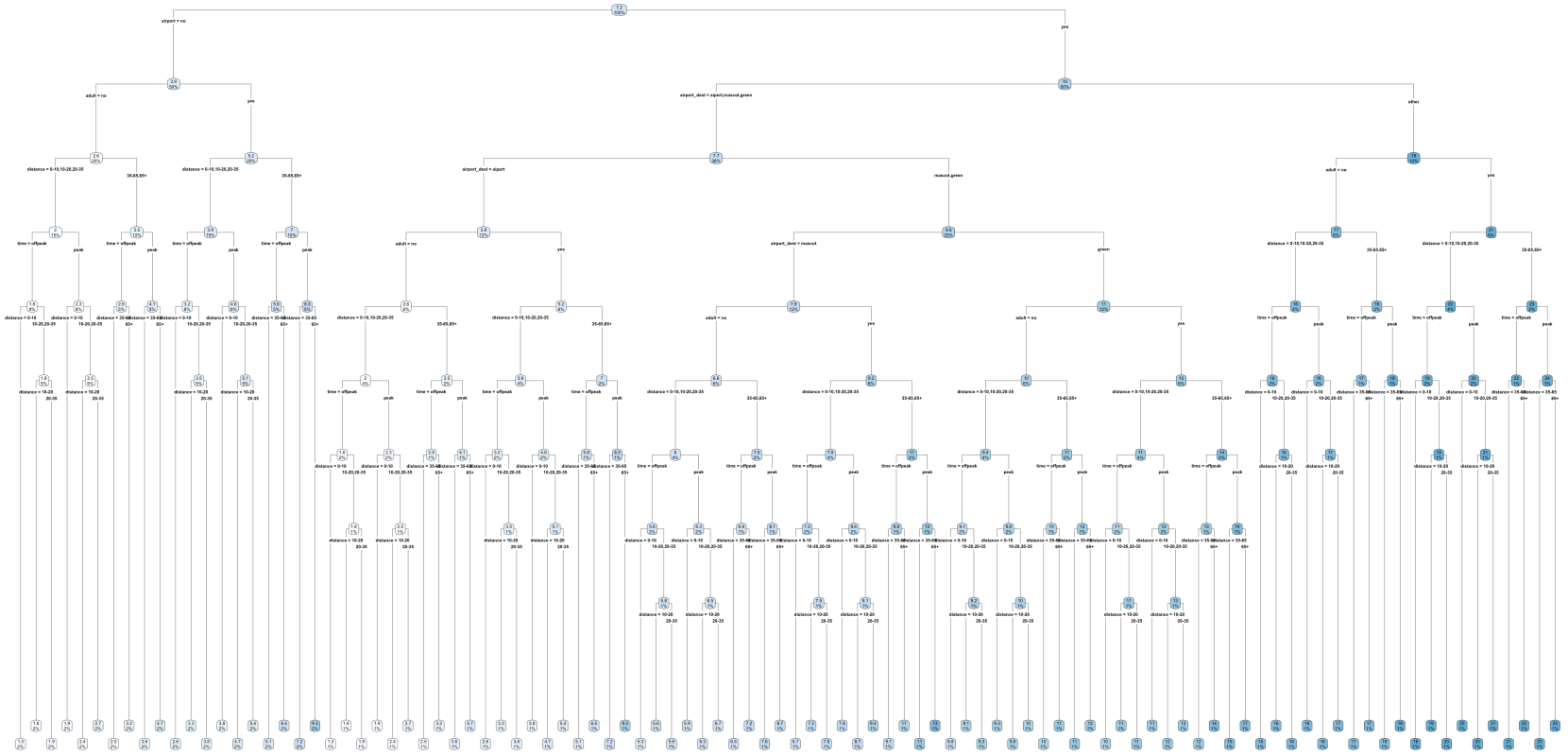
Black box machine learning models are currently being used for high-stakes decision making throughout society, causing problems in healthcare, criminal justice and other domains. Some people hope that creating methods for explaining these black box models will alleviate some of the problems, but trying to explain black box models, rather than creating models that are interpretable in the first place, is likely to perpetuate bad practice and can potentially cause great harm to society. The way forward is to design models that are inherently interpretable. This Perspective clarifies the chasm between explaining black boxes and using inherently interpretable models, outlines several key reasons why explainable black boxes should be avoided in high-stakes decisions, identifies challenges to interpretable machine learning, and provides several example applications where interpretable models could potentially replace black box models in criminal justice, healthcare and computer vision.

Trees are interpretable!



Train prices

Trees are interpretable?



Full train pricing

Linear models & LocalGLMNet

A GLM has the form

$$\hat{y} = g^{-1}\left(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p\right)$$

where $\beta_0, \ldots, \beta_p$ are the model parameters.

Global & local interpretations are easy to obtain.

LocalGLMNet extends this to a neural network.

$$\hat{y}_i = g^{-1}\left(\beta_0(\boldsymbol{x}_i) + \beta_1(\boldsymbol{x}_i) x_{i1} + \cdots + \beta_p(\boldsymbol{x}_i) x_{ip}\right)$$

A GLM with local parameters $\beta_0(\boldsymbol{x}_i), \ldots, \beta_p(\boldsymbol{x}_i)$ for each observation $\boldsymbol{x}_i$. The local parameters are the output of a neural network.

Source: Richman and Wüthrich (2023), *LocalGLMnet: interpretable deep learning for tabular data*, Scandinavian Actuarial Journal (2023.1), pp. 71–95.

Lecture Outline

- Interpretability
- Inherent Interpretability
- **Post-hoc Interpretability**
- Illustrative Example
- Illustrative Example (Fixed)

Permutation importance

- Inputs: fitted model m, tabular dataset D.
- Compute the reference score s of the model m on data D (for instance the accuracy for a classifier or the R^2 for a regressor).
- For each feature j (column of D):
  - For each repetition k in $1, \ldots, K$:
    - Randomly shuffle column j of dataset D to generate a corrupted version of the data named $\tilde{D}_{k,j}$.
    - Compute the score $s_{k,j}$ of model m on corrupted data $\tilde{D}_{k,j}$.
  - Compute importance i_j for feature f_j defined as:

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

Originally proposed by Breiman (2001), *Random forests*, Machine learning (45), pp. 5-32.

Extended by Fisher et al. (2019), *All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously*, Journal of Machine Learning Research (20.177), pp. 1-81.

Source: scikit-learn documentation, permutation_importance function.

Permutation importance

```
def permutation_test(model, X, y, num_reps=1, seed=42):
    """
    Run the permutation test for variable importance.
    Returns matrix of shape (X.shape[1], len(model.evaluate(X, y))).
    """
    rnd.seed(seed)
    scores = []

    for j in range(X.shape[1]):
        original_column = np.copy(X[:, j])
        col_scores = []

        for r in range(num_reps):
            rnd.shuffle(X[:,j])
            col_scores.append(model.evaluate(X, y, verbose=0))

        scores.append(np.mean(col_scores, axis=0))
        X[:,j] = original_column

    return np.array(scores)
```

LIME

Local Interpretable Model-agnostic Explanations employs an interpretable surrogate model to explain locally how the black-box model makes predictions for individual instances.

E.g. a black-box model predicts Bob's premium as the highest among all policyholders. LIME uses an interpretable model (a linear regression) to explain how Bob's features influence the black-box model's prediction.

See "Why Should I Trust You?": Explaining the Predictions of Any Classifier.

Globally vs. Locally Faithful

Globally Faithful

The interpretable model's explanations accurately reflect the behaviour of the black-box model across the entire input space.

Locally Faithful

The interpretable model's explanations accurately reflect the behaviour of the black-box model for a specific instance.

LIME aims to construct an interpretable model that mimics the black-box model's behaviour in a *locally faithful* manner.

LIME Algorithm

Suppose we want to explain the instance $\boldsymbol{x}_{\text{Bob}} = (1, 2, 0.5)$.

1. Generate perturbed examples of $\boldsymbol{x}_{\text{Bob}}$ and use the trained gamma MDN f to make predictions:

$$\begin{aligned} \boldsymbol{x}'^{(1)}_{\text{Bob}} &= (1.1, 1.9, 0.6), \quad & f\left(\boldsymbol{x}'^{(1)}_{\text{Bob}}\right) &= 34000 \\ \boldsymbol{x}'^{(2)}_{\text{Bob}} &= (0.8, 2.1, 0.4), \quad & f\left(\boldsymbol{x}'^{(2)}_{\text{Bob}}\right) &= 31000 \\ &\vdots & &\vdots \end{aligned}$$

   We can then construct a dataset of N_{Examples} perturbed examples: $\mathcal{D}_{\text{LIME}} = \left(\left\{\boldsymbol{x}'^{(i)}_{\text{Bob}}, f\left(\boldsymbol{x}'^{(i)}_{\text{Bob}}\right)\right\}\right)_{i=0}^{N_{\text{Examples}}}$.

UNSW SYDNEY

LIME Algorithm

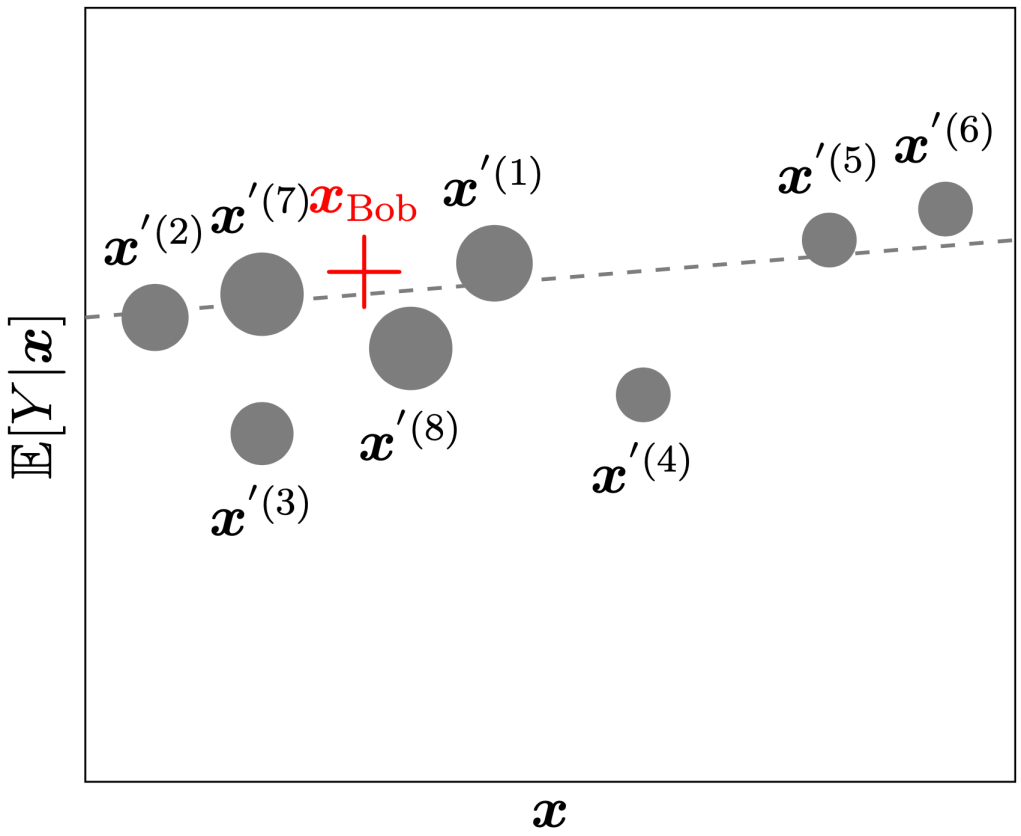
2. Fit an interpretable model g, i.e., a linear regression using $\mathcal{D}_{\text{LIME}}$ and the following loss function:

$$\mathcal{L}_{\text{LIME}}(f, g, \pi_{\boldsymbol{x}_{\text{Bob}}}) = \sum_{i=1}^{N_{\text{Examples}}} \pi_{\boldsymbol{x}_{\text{Bob}}}\left(\boldsymbol{x}_{\text{Bob}}^{'(i)}\right) \cdot \left(f\left(\boldsymbol{x}_{\text{Bob}}^{'(i)}\right) - g\left(\boldsymbol{x}_{\text{Bob}}^{'(i)}\right)\right)^2,$$

where $\pi_{\boldsymbol{x}_{\text{Bob}}}\left(\boldsymbol{x}_{\text{Bob}}^{'(i)}\right)$ represents the distance from the perturbed example $\boldsymbol{x}_{\text{Bob}}^{'(i)}$ to the instance to be explained $\boldsymbol{x}_{\text{Bob}}$.

UNSW SYDNEY

“Explaining” to Bob



The bold red cross is the instance being explained. LIME samples instances (grey nodes), gets predictions using f (gamma MDN) and weighs them by the proximity to the instance being explained (represented here by size). The dashed line g is the learned local explanation.

SHAP Values

The SHapley Additive exPlanations (SHAP) value helps to quantify the contribution of each feature to the prediction for a specific instance.

The SHAP value for the jth feature is defined as

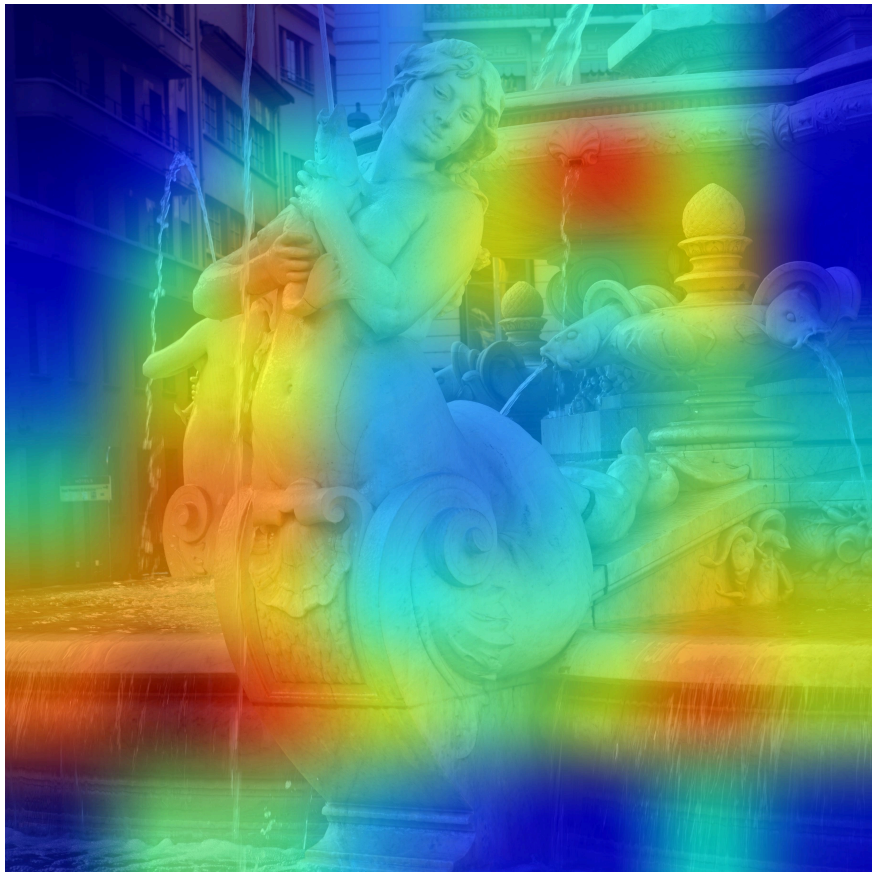
$$\text{SHAP}^{(j)}(\boldsymbol{x}) = \sum_{U \subset \{1,\ldots,p\} \setminus \{j\}} \frac{1}{p} \binom{p-1}{|U|}^{-1} \left(\mathbb{E}[Y|\boldsymbol{x}^{(U \cup \{j\})}] - \mathbb{E}[Y|\boldsymbol{x}^{(U)}] \right),$$

where p is the number of features. A positive SHAP value indicates that the variable increases the prediction value.

Reference: Lundberg & Lee (2017), A Unified Approach to Interpreting Model Predictions, Advances in Neural Information Processing Systems, 30.

Grad-CAM

Original image



Grad-CAM

See, e.g., Keras tutorial.

See Chollet (2021), Deep Learning with Python, Section 9.4.3.

Criticism

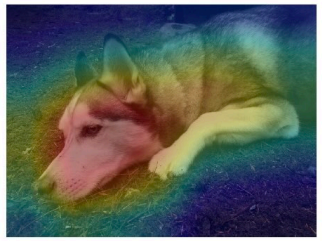
	Test Image	Evidence for Animal Being a Siberian Husky	Evidence for Animal Being a Transverse Flute
Explanations Using Attention Maps			

Figure 2: Saliency does not explain anything except where the network is looking. We have no idea why this image is labeled as either a dog or a musical instrument when considering only saliency. The explanations look essentially the same for both classes. Figure credit: Chaofan Chen and [28].

Multiple conflicting explanations.

Source: Rudin (2019), Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nature Machine Intelligence.

Lecture Outline

- Interpretability
- Inherent Interpretability
- Post-hoc Interpretability
- **Illustrative Example**
- Illustrative Example (Fixed)

First attempt at NLP task

```
1 df_raw["SUMMARY_EN"]
```

```
0       V1, a 2000 Pontiac Montana minivan,
made a lef ...
1       The crash occurred in the eastbound
lane of a  ...
2       This crash occurred just after the
noon time h ...
                                  ...
6946    The crash occurred in the eastbound
lanes of a ...
6947    This single-vehicle crash occurred in
a rural  ...
6948    This two vehicle daytime collision
occurred mi ...
Name: SUMMARY_EN, Length: 6949, dtype: object
```

```
1 df_raw["NUM_VEHICLES"].value_counts()\
2   .sort_index()
```

```
NUM_VEHICLES
1     1822
2     4151
3+     976
Name: count, dtype: int64
```

Source: JSchelldorfer's GitHub.

Bag of words for the top 1,000 words

```
vect = CountVectorizer(max_features=1_000, stop_words="english")
vect.fit(X_train["SUMMARY_EN"])

X_train_bow = vectorise_dataset(X_train, vect)
X_val_bow = vectorise_dataset(X_val, vect)
X_test_bow = vectorise_dataset(X_test, vect)

vectorise_dataset(X_train, vect, dataframe=True).head()
```

	10	105	113	12	15	150	16	17	18	180	...	yield	201
2532	0	0	0	0	0	0	0	0	0	0	...	0	0
6209	0	0	0	0	0	0	0	0	0	0	...	0	0
2561	1	0	1	0	0	0	0	0	0	0	...	0	0
6664	0	0	0	0	0	0	0	0	0	0	...	0	0
4214	0	0	0	0	0	0	0	0	0	0	...	0	0

5 rows × 1008 columns

UNSW SYDNEY

Trained a basic neural network on that

```
num_features = X_train_bow.shape[1]
num_cats = df_raw["NUM_VEHICLES"].nunique()
model = build_model(num_features, num_cats)
es = EarlyStopping(patience=1, restore_best_weights=True, monitor="val_accuracy")
```

```
model.fit(X_train_bow, y_train, epochs=10,
    callbacks=[es], validation_data=(X_val_bow, y_val), verbose=0)
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 100)	100,900
dense_1 (Dense)	(None, 3)	303

Total params: 303,611 (1.16 MB)
Trainable params: 101,203 (395.32 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 202,408 (790.66 KB)

```
model.evaluate(X_train_bow, y_train, verbose=0)
```

```
[0.0044780503958463 67, 1.0, 1.0]
```

```
model.evaluate(X_val_bow, y_val, verbose=0)
```

```
[8.163677215576172, 0.9784172773361206, 0.9985611438751221]
```

UNSW SYDNEY

Permutation importance algorithm

Taken directly from scikit-learn documentation:

- Inputs: fitted predictive model m, tabular dataset (training or validation) D.
- Compute the reference score s of the model m on data D (for instance the accuracy for a classifier or the R^2 for a regressor).
- For each feature j (column of D):
  - For each repetition k in $1, \ldots, K$:
    - Randomly shuffle column j of dataset D to generate a corrupted version of the data named $\tilde{D}_{k,j}$.
    - Compute the score $s_{k,j}$ of model m on corrupted data $\tilde{D}_{k,j}$.
  - Compute importance i_j for feature f_j defined as:

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

Source: scikit-learn documentation, permutation_importance function.

Find important inputs

```python
def permutation_test(model, X, y, num_reps=1, seed=42):
    """
    Run the permutation test for variable importance.
    Returns matrix of shape (X.shape[1], len(model.evaluate(X, y))).
    """
    rnd.seed(seed)
    scores = []

    for j in range(X.shape[1]):
        original_column = np.copy(X[:, j])
        col_scores = []

        for r in range(num_reps):
            rnd.shuffle(X[:,j])
            col_scores.append(model.evaluate(X, y, verbose=0))

        scores.append(np.mean(col_scores, axis=0))
        X[:,j] = original_column

    return np.array(scores)
```

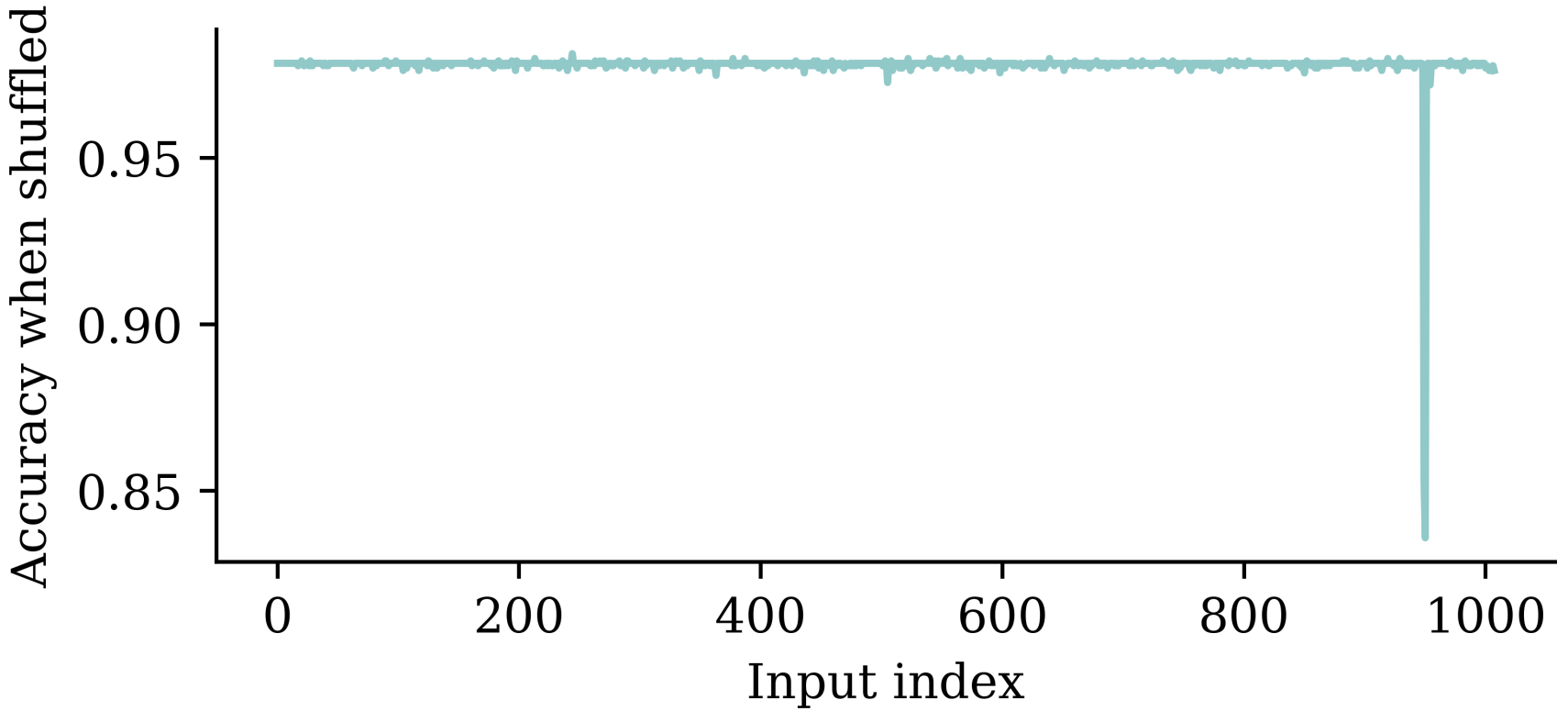
Run the permutation test

```
1 all_perm_scores = permutation_test(model, X_val_bow, y_val)
2 all_perm_scores
```

```
array([[ 8.16,  0.98,  1.  ],
       [ 8.16,  0.98,  1.  ],
       [ 8.16,  0.98,  1.  ],
       ...,
       [ 8.48,  0.98,  1.  ],
       [ 8.47,  0.98,  1.  ],
       [14.59,  0.98,  1.  ]])
```

Plot the permutated accuracies

```
perm_scores = all_perm_scores[:,1]
plt.plot(perm_scores)
plt.xlabel("Input index")
plt.ylabel("Accuracy when shuffled");
```



Find the most significant inputs

```
vocab = vect.get_feature_names_out()
input_cols = list(vocab) + weather_cols

best_input_inds = np.argsort(perm_scores)[:100]
best_inputs = [input_cols[idx] for idx in best_input_inds]

print(best_inputs)
```

```
['v3', 'v2', 'vehicle', 'involved', 'event', 'harmful', 'stated', 'motor', 'WEATHER8',
'left', 'v1', 'just', 'turn', 'traveling', 'approximately', 'medication', 'hurry',
'encroaching', 'chevrolet', 'parked', 'continued', 'saw', 'road', 'rest', 'distraction',
'ahead', 'wet', 'hit', 'WEATHER5', 'WEATHER6', 'WEATHER4', 'old', 'v4', '48', 'rested',
'direction', 'occurred', 'kph', 'clear', 'right', 'miles', 'uphill', 'WEATHER3', 'denied',
'80', 'attempt', 'thinking', 'assumption', 'damage', 'runs', 'time', 'consisted',
'following', 'towed', 'WEATHER1', 'alcohol', 'mph', 'pickup', 'lane', 'conversing', 'make',
'started', 'maneuver', 'stopped', 'store', 'car', 'local', 'dry', 'median', 'south',
'driver', 'higher', 'pre', 'northbound', 'impacting', 'congested', 'health', 'southwest',
'gmc', 'observed', 'parking', 'partially', 'heart', 'shoulders', 'shoulder', 'southeast',
'came', 'heard', 'gap', 'southbound', 'conditions', 'contacted', 'change', 'compact', '2002',
'cell', 'causing', 'oldsmobile', 'half', 'hard']
```

UNSW SYDNEY

How about a simple decision tree?

```
from sklearn import tree

clf = tree.DecisionTreeClassifier(random_state=0, max_leaf_nodes=3)
clf.fit(X_train_bow[:, best_input_inds], y_train);
```

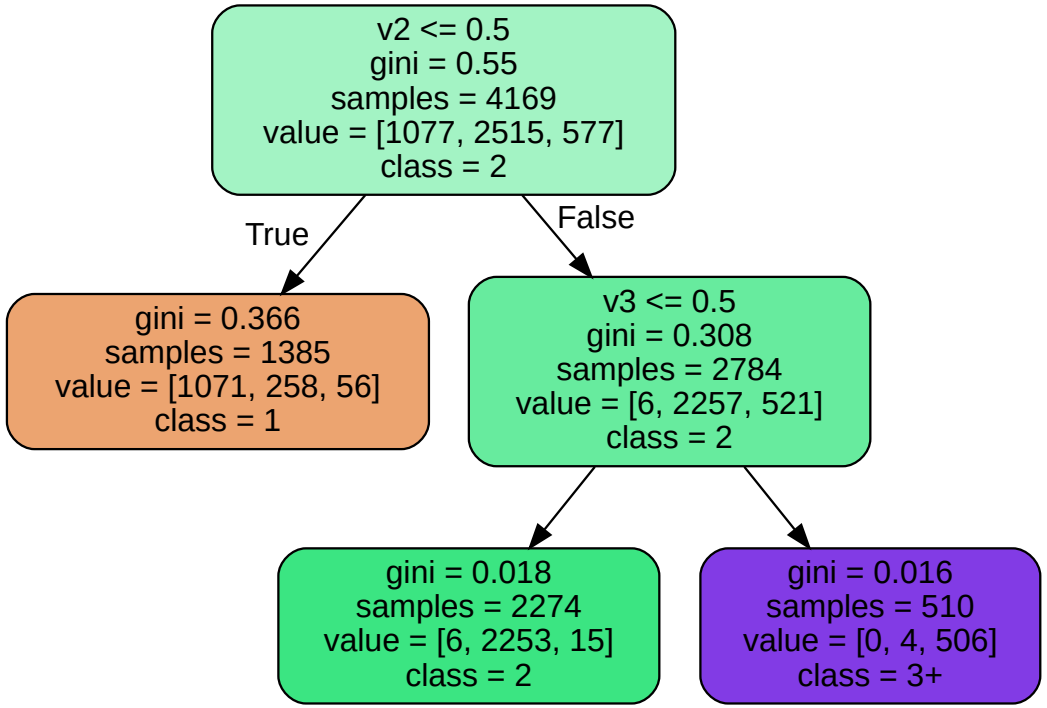
```
print(clf.score(X_train_bow[:, best_input_inds], y_train))
print(clf.score(X_val_bow[:, best_input_inds], y_val))
```

```
0.9186855360997841
0.9316546762589928
```

The decision tree ends up giving pretty good results.

Decision tree

```
1 tree.plot_tree(clf, feature_names=best_inputs, filled=True);
```




```
1 print(np.where(clf.feature_importances_ > 0)[0])
2 [best_inputs[ind] for ind in np.where(clf.feature_importances_ > 0)[0]]
```

```
[0 1]
['v3', 'v2']
```

UNSW SYDNEY

Lecture Outline

- Interpretability
- Inherent Interpretability
- Post-hoc Interpretability
- Illustrative Example
- **Illustrative Example (Fixed)**

This is why we replace “v1”, “v2”, “v3”

There was a slide in the NLP deck titled “Just ignore this for now…” That was going through each summary and replacing the words “V1”, “V2”, “V3” with random numbers. This was done to see if the model was overfitting to these words.

```
model.fit(X_train_bow, y_train, epochs=10,
    callbacks=[es], validation_data=(X_val_bow, y_val), verbose=0);
```

Retraining on the fixed dataset gives us a more realistic (lower) accuracy.

```
model.evaluate(X_train_bow, y_train, verbose=0)
```

```
[0.07491350173950195, 0.986567497253418, 0.9997601509094238]
```

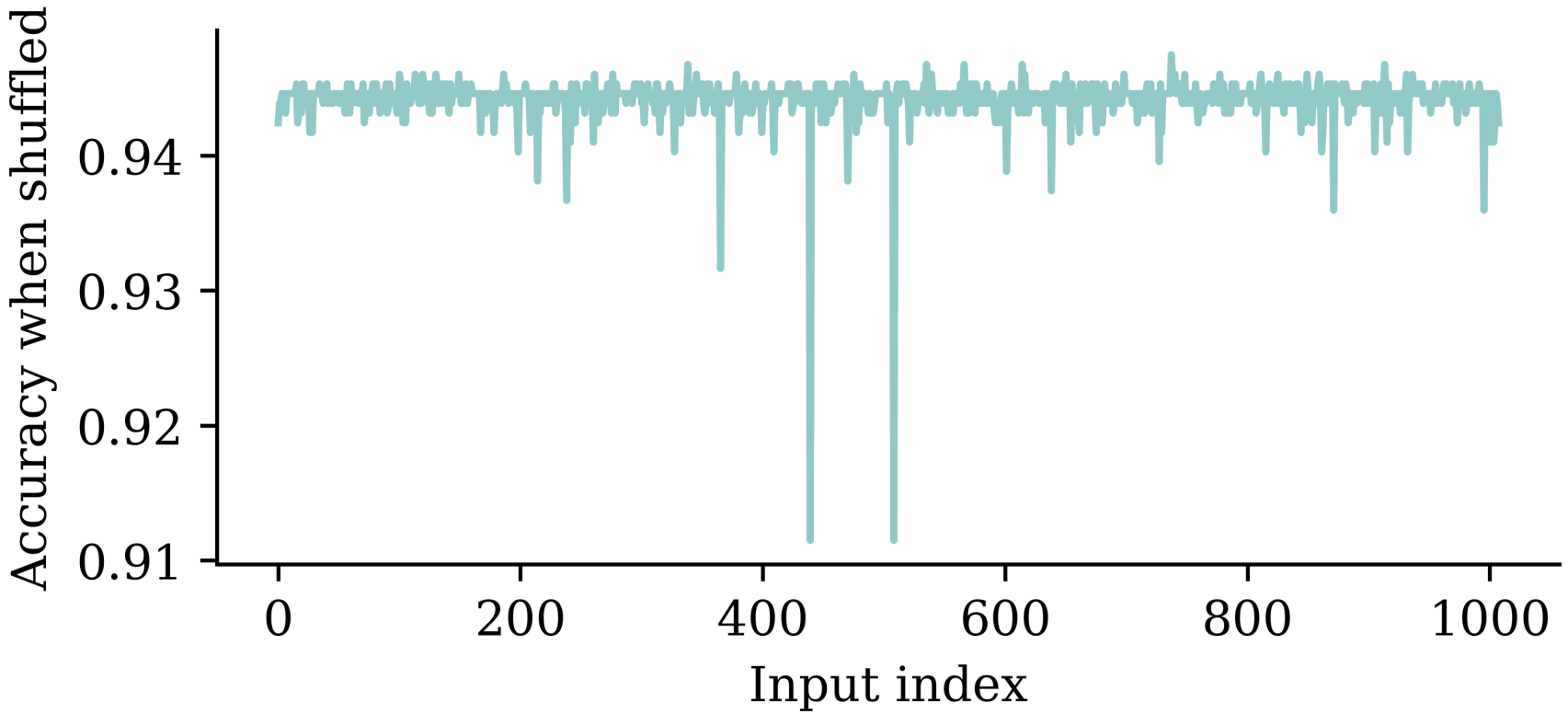
```
model.evaluate(X_val_bow, y_val, verbose=0)
```

```
[2.865464925765991, 0.9446043372154236, 0.9956834316253662]
```

UNSW SYDNEY

Permutation importance accuracy plot

```
1 perm_scores = permutation_test(model, X_val_bow, y_val)[:,1]
2 plt.plot(perm_scores)
3 plt.xlabel("Input index"); plt.ylabel("Accuracy when shuffled");
```



Find the most significant inputs

```
1 vocab = vect.get_feature_names_out()
2 input_cols = list(vocab) + weather_cols
3
4 best_input_inds = np.argsort(perm_scores)[:100]
5 best_inputs = [input_cols[idx] for idx in best_input_inds]
6
7 print(best_inputs)
```

```
['harmful', 'involved', 'event', 'year', 'struck', 'contacted', 'old', 'impact', 'coded',
'motor', 'rear', 'towed', 'stop', 'driven', 'turning', 'single', 'forward', 'chevrolet',
'lanes', 'crash', 'parked', 'continued', 'WEATHER4', 'WEATHER1', 'travel', 'divided',
'brakes', 'include', 'came', 'stopped', 'final', 'factor', 'clear', '2004', 'speed', '2002',
'reason', 'passing', 'pickup', 'crossing', 'driving', 'ahead', 'did', 'control', '10',
'highway', 'WEATHER5', 'weather', 'traveling', 'surveillance', 'stated', 'spin', 'heart',
'road', 'pole', 'occupants', 'afternoon', 'moderate', 'mirror', 'including', 'pushed',
'55mph', 'intersection', 'WEATHER8', '1993', 'cross', 'curve', 'slight', 'dark', 'day',
'sight', 'distance', 'seat', 'saw', 'said', 'drivers', 'earlier', 'early', 'dodge',
'corrected', 'state', 'contributed', '16', 'trying', 'trip', 'buick', 'trailer', 'traffic',
'tractor', 'cherokee', '42', 'taken', '41', 'cloudy', 'collision', 'congested', 'contact',
'eastbound', 'roadways', 'roadway']
```

UNSW SYDNEY

How about a simple decision tree?

```
clf = tree.DecisionTreeClassifier(random_state=0, max_leaf_nodes=3)
clf.fit(X_train_bow[:, best_input_inds], y_train);
```

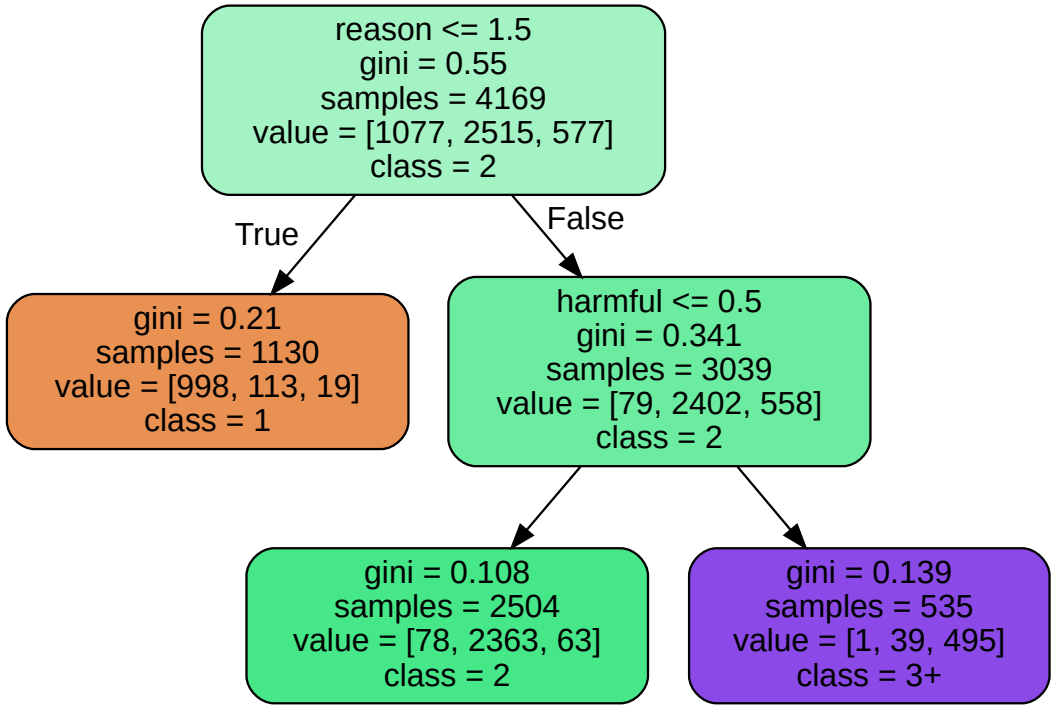
```
print(clf.score(X_train_bow[:, best_input_inds], y_train))
print(clf.score(X_val_bow[:, best_input_inds], y_val))
```

```
0.9249220436555529
0.9381294964028777
```

UNSW SYDNEY

Decision tree

```
1 tree.plot_tree(clf, feature_names=best_inputs, filled=True);
```




```
1 print(np.where(clf.feature_importances_ > 0)[0])
2 [best_inputs[ind] for ind in np.where(clf.feature_importances_ > 0)[0]]
```

```
[ 0 36]
['harmful', 'reason']
```

UNSW SYDNEY

Package Versions

```
1 from watermark import watermark
2 print(watermark(python=True, packages="keras,matplotlib,numpy,pandas,seaborn,scipy,torch
```

```
Python implementation: CPython
Python version       : 3.11.11
IPython version      : 8.32.0

keras     : 3.8.0
matplotlib: 3.10.0
numpy     : 1.26.4
pandas    : 2.2.3
seaborn   : 0.13.2
scipy     : 1.13.1
torch     : 2.5.1+cu124
tensorflow: 2.18.0
tf_keras  : 2.18.0
```

UNSW SYDNEY

Glossary

- global interpretability
- Grad-CAM
- inherent interpretability
- LIME
- local interpretability
- permutation importance
- post-hoc interpretability
- SHAP values