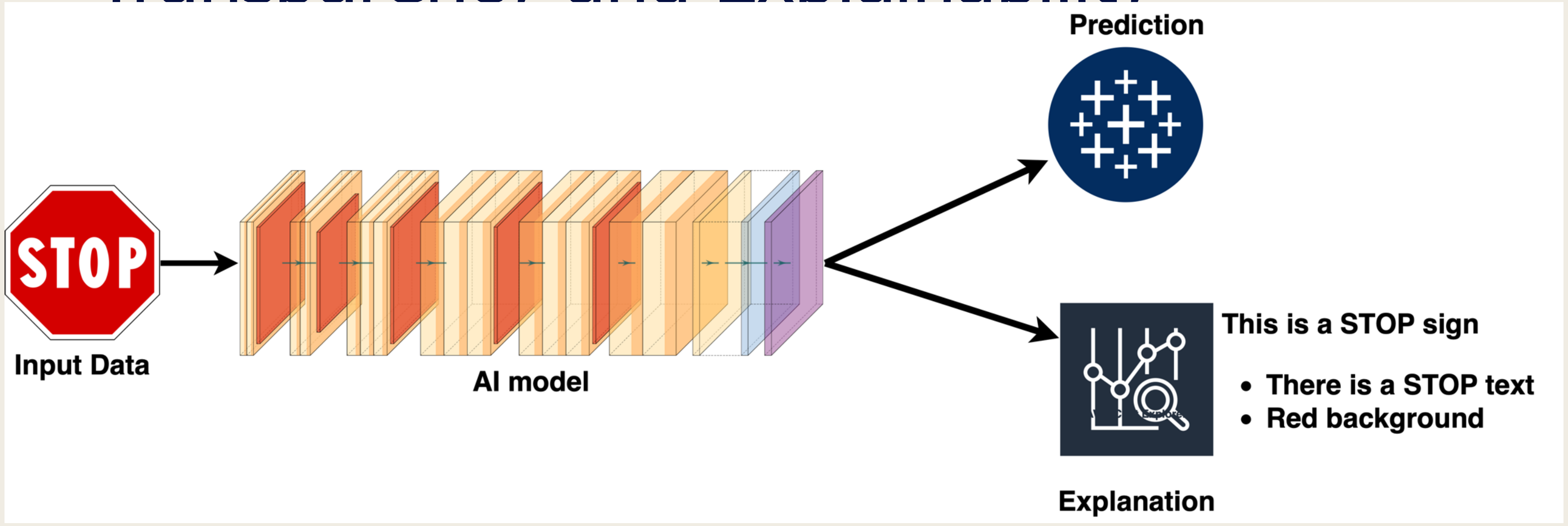
University of Stavanger

F. Ozgur Catak – f.ozgur.Catak@uis.no

DAT945: XAI

Transparency and Explainability



- Not just to give you an answer, but to give you an answer and explain how it reached that conclusion.
- If it says, "that's a stop sign," you can ask, "Why do you think it's a stop sign?" And AI will break it down for you, saying something like, "I saw the red color and the shape, just like the stop signs I learned from."

Types of Explainability Approaches

- **Rule-based Explainability:** Uses decision trees and rules to provide transparent predictions.
- **Feature Importance:** Techniques like LIME and SHAP identify key features influencing predictions.
- **Model-specific Approaches:** Leveraging unique characteristics of models like DNNs, SVMs.
- **Model-agnostic Approaches:** Explain decisions regardless of model type.

Local Interpretable Model-agnostic Explanations (LIME)

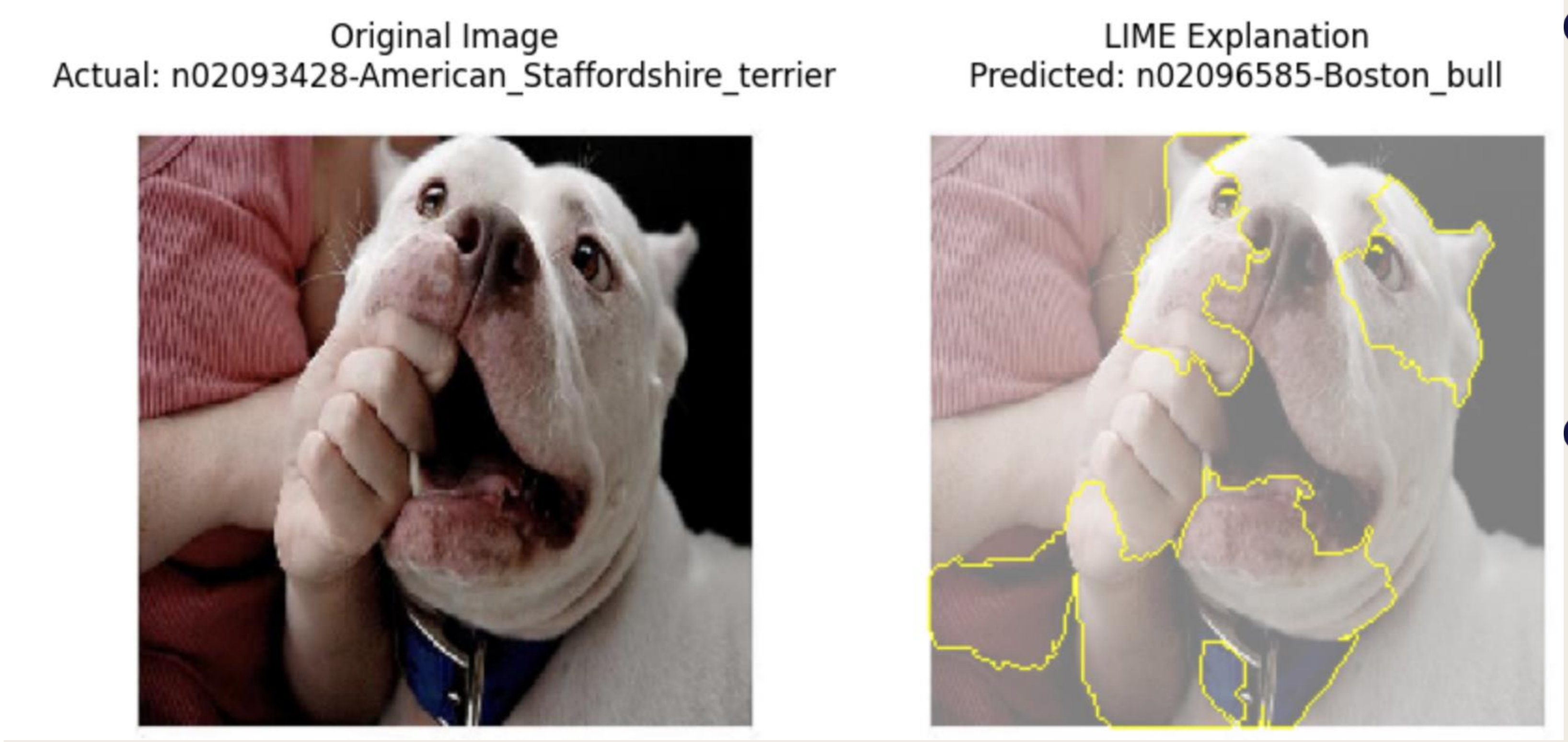
- Generates interpretable models to approximate complex models locally.

- Generates a number of perturbed samples by making small changes to the selected instance.
  - The nature of these perturbations depends on the type of data (tabular, text, or image).

- Steps: Sampling, Prediction, Weighting, Model Approximation, Explanation.

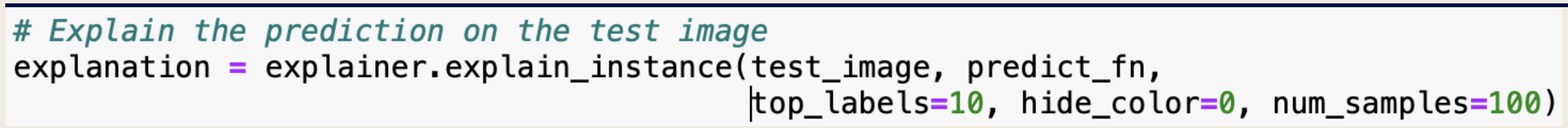
- Visual interpretation of model predictions.

Local Interpretable Model-agnostic Explanations (LIME)


Original Image
Actual: n02093428-American_Staffordshire_terrier
LIME Explanation
Predicted: n02096585-Boston_bull

- The yellow highlighted regions indicate the parts of the image that the model focused on when making its prediction.
- These regions are considered significant by the model in deciding that the dog is a Boston Bull instead of an American Staffordshire Terrier.

Local Interpretable Model-agnostic Explanations (LIME)



```
# Explain the prediction on the test image
explanation = explainer.explain_instance(test_image, predict_fn,
                                         top_labels=10, hide_color=0, num_samples=100)
```

- **explainer.explain_instance()**:
  - This function generates explanations for a single instance (in this case, test_image).
- **test_image**:
  - The specific instance (image) for which we want to understand the model's prediction. This is the input data for which the explanation is generated.
- **predict_fn**:
  - A function that takes a batch of input instances (such as images) and returns the model's predictions. This function is used by LIME to get the prediction for the perturbed instances it creates.
- **top_labels=10**:
  - This parameter specifies that the explanation should focus on the top 10 predicted labels. LIME will generate explanations for the predictions with the highest probabilities.
- **hide_color=0**:
  - The color used to hide parts of the image when generating perturbations. Typically, setting this to 0 hides parts of the image by making those parts black.
- **num_samples=100**:
  - The number of perturbed samples to generate. LIME creates several variations of the original image by hiding different parts of it and then uses these samples to understand which parts of the image are most important for the model's prediction.

Local Interpretable Model-agnostic Explanations (LIME)

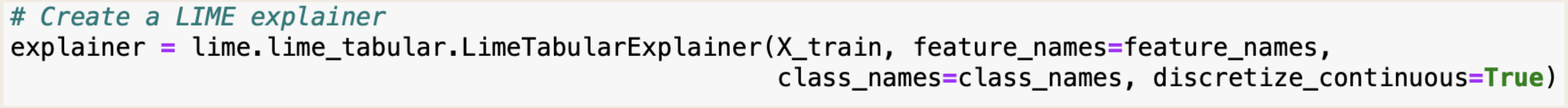
```python
# Get the explanation for the top predicted class
temp, mask = explanation.get_image_and_mask(explanation.top_labels[0],
                                            positive_only=True, num_features=10, hide_rest=False)
```

- **explanation.get_image_and_mask()**:
  - This method is part of the LIME (Local Interpretable Model-agnostic Explanations) library.
  - It is used to retrieve an image with highlighted regions (mask) that show which parts of the image contributed the most to the model's prediction.
- **explanation.top_labels[0]**:
  - explanation.top_labels is a list of class labels that the model predicted for the given instance, sorted by their prediction probabilities in descending order.
  - explanation.top_labels[0] refers to the class with the highest predicted probability. This is the class for which we want to understand the model's decision.

Parameters:

- **positive_only=True**:
  - When set to True, only the features that positively contribute to the prediction of the class are highlighted. Negative contributions are ignored.
- **num_features=10**:
  - Specifies the number of top features to be included in the explanation. In this case, it will highlight the top 10 features (regions) that most contribute to the prediction.
- **hide_rest=False**:
  - When set to False, the entire image is shown, including areas that are not part of the explanation. If set to True, only the highlighted areas would be visible, and the rest of the image would be hidden.

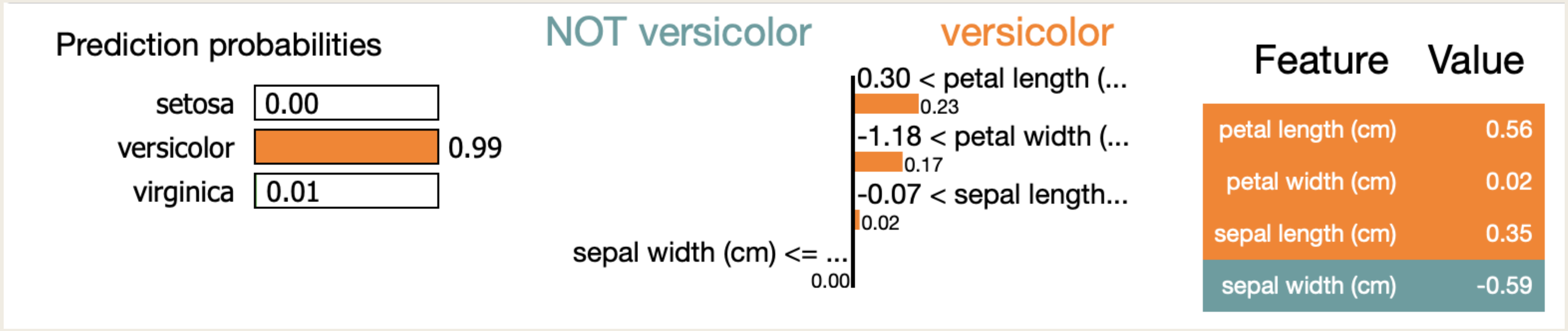
LIME with Tabular Data



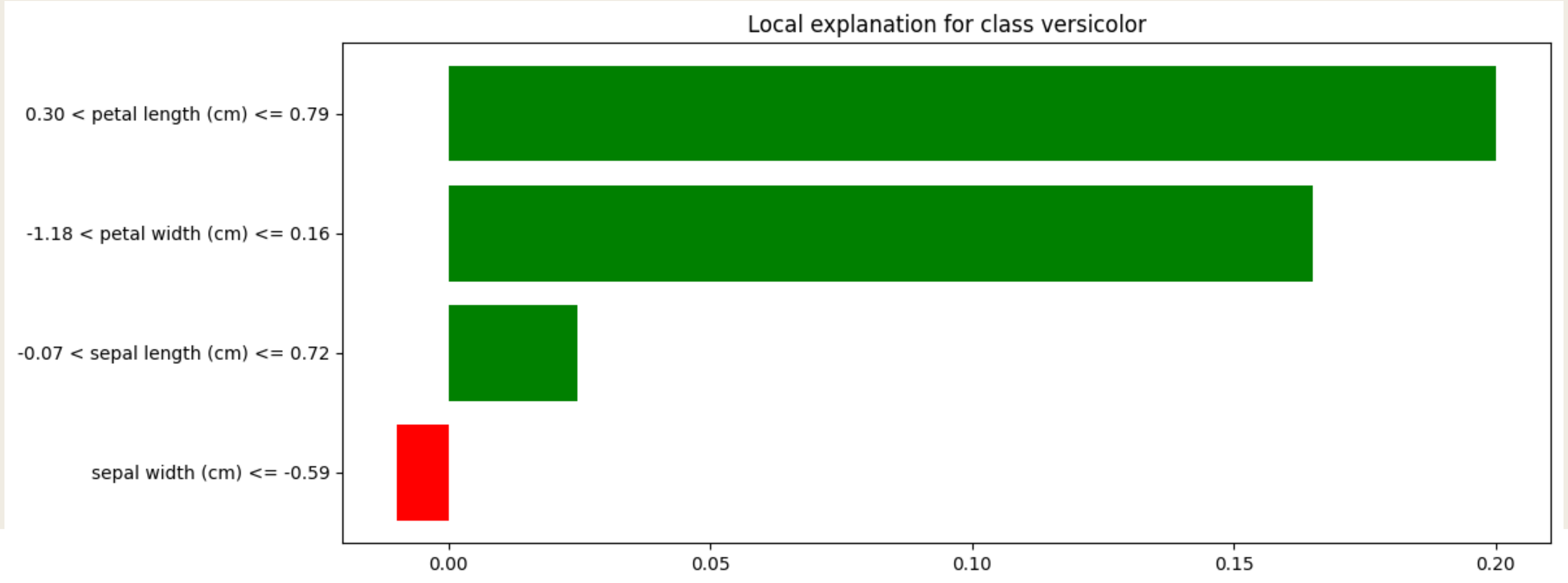
```python
# Create a LIME explainer
explainer = lime.lime_tabular.LimeTabularExplainer(X_train, feature_names=feature_names,
                                                   class_names=class_names, discretize_continuous=True)
```

- **X_train**: The training dataset. This dataset is used to understand the distribution of the features and the relationships between them. It should be a 2D numpy array or a pandas DataFrame where rows are samples and columns are features.
- **feature_names**: A list of names corresponding to the features (columns) in X_train. This is used to make the explanations more interpretable and human-readable.
- **class_names**: A list of class names for the target variable. This helps in providing meaningful labels in the explanations for classification problems.
- **discretize_continuous=True**: This parameter specifies whether continuous features should be discretized into bins. Discretization helps in making the model's explanations more interpretable, especially for non-linear models. When set to True, continuous features are divided into a specified number of bins, and these bins are treated as categorical features.

LIME with Tabular Data



Feature	Value
petal length (cm)	0.56
petal width (cm)	0.02
sepal length (cm)	0.35
sepal width (cm)	-0.59



SHAP (SHapley Additive exPlanations)

- A popular method for explaining the output of machine learning models.
- Based on cooperative game theory and provides a unified framework for interpreting predictions.
- SHAP values are a way to assign each feature in an input data point a contribution to the prediction made by the model.

LIME vs SHAP

1. **Underlying Theory:**

   1. **LIME**: Based on the idea of locally approximating the model with a simpler, interpretable model (e.g., linear regression). It perturbs the input data and observes the model's output to learn a local surrogate model.

   2. **SHAP**: Based on Shapley values from cooperative game theory. It assigns each feature a contribution to the prediction by considering all possible feature combinations.

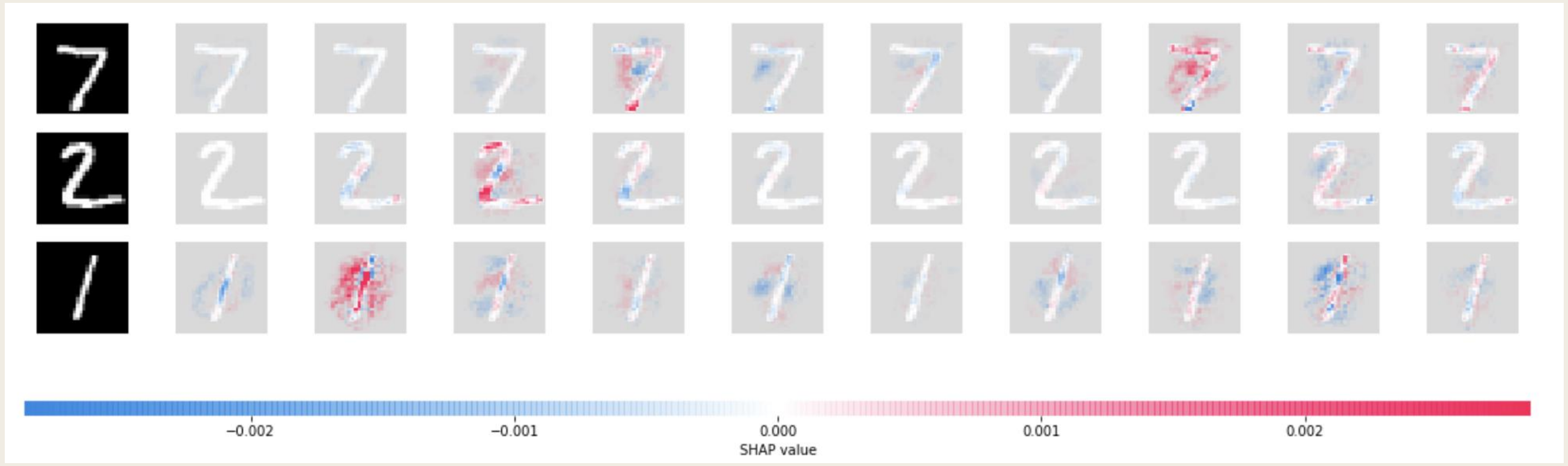
1. **Speed and Computation**:
   1. **LIME**: Generally faster for local explanations since it only needs to perturb the data locally and fit a simple model.

   2. **SHAP**: Can be computationally expensive

SHAP

```
# Create a SHAP explainer
explainer = shap.GradientExplainer(model, test_images[:100])
```

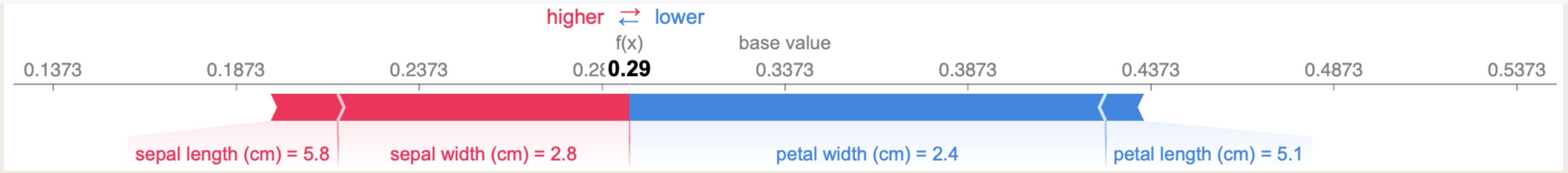
```
# Generate SHAP values for the selected image
shap_values = explainer.shap_values(np.expand_dims(test_image, axis=0))

# Plot the SHAP values
shap.image_plot(shap_values, np.expand_dims(test_image, axis=0))
```

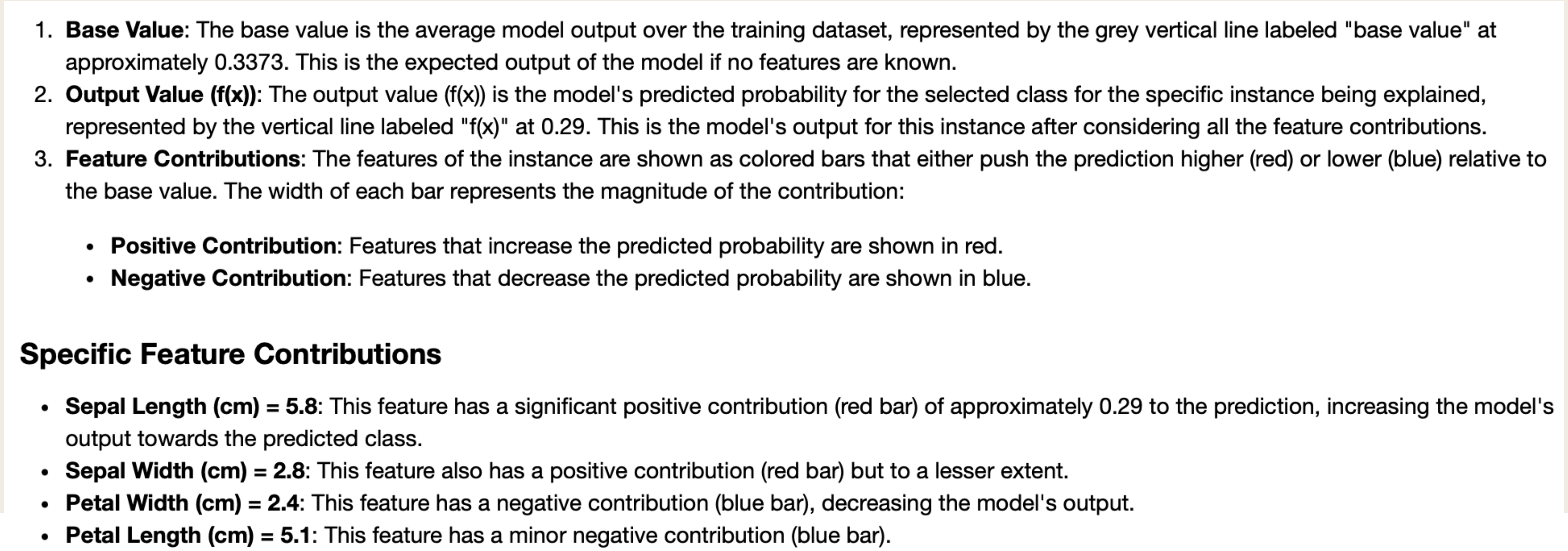


Tabular Data with SHAP

- RandomForest classifier for Iris dataset

```
explainer = shap.KernelExplainer(model.predict_proba, X_train)
shap_values = explainer.shap_values(X_test.iloc[0, :])
shap.force_plot(explainer.expected_value[0], shap_values[:, 0], X_test.iloc[0, :])
```





1. **Base Value**: The base value is the average model output over the training dataset, represented by the grey vertical line labeled "base value" at approximately 0.3373. This is the expected output of the model if no features are known.
2. **Output Value (f(x))**: The output value (f(x)) is the model's predicted probability for the selected class for the specific instance being explained, represented by the vertical line labeled "f(x)" at 0.29. This is the model's output for this instance after considering all the feature contributions.
3. **Feature Contributions**: The features of the instance are shown as colored bars that either push the prediction higher (red) or lower (blue) relative to the base value. The width of each bar represents the magnitude of the contribution:
   - **Positive Contribution**: Features that increase the predicted probability are shown in red.
   - **Negative Contribution**: Features that decrease the predicted probability are shown in blue.

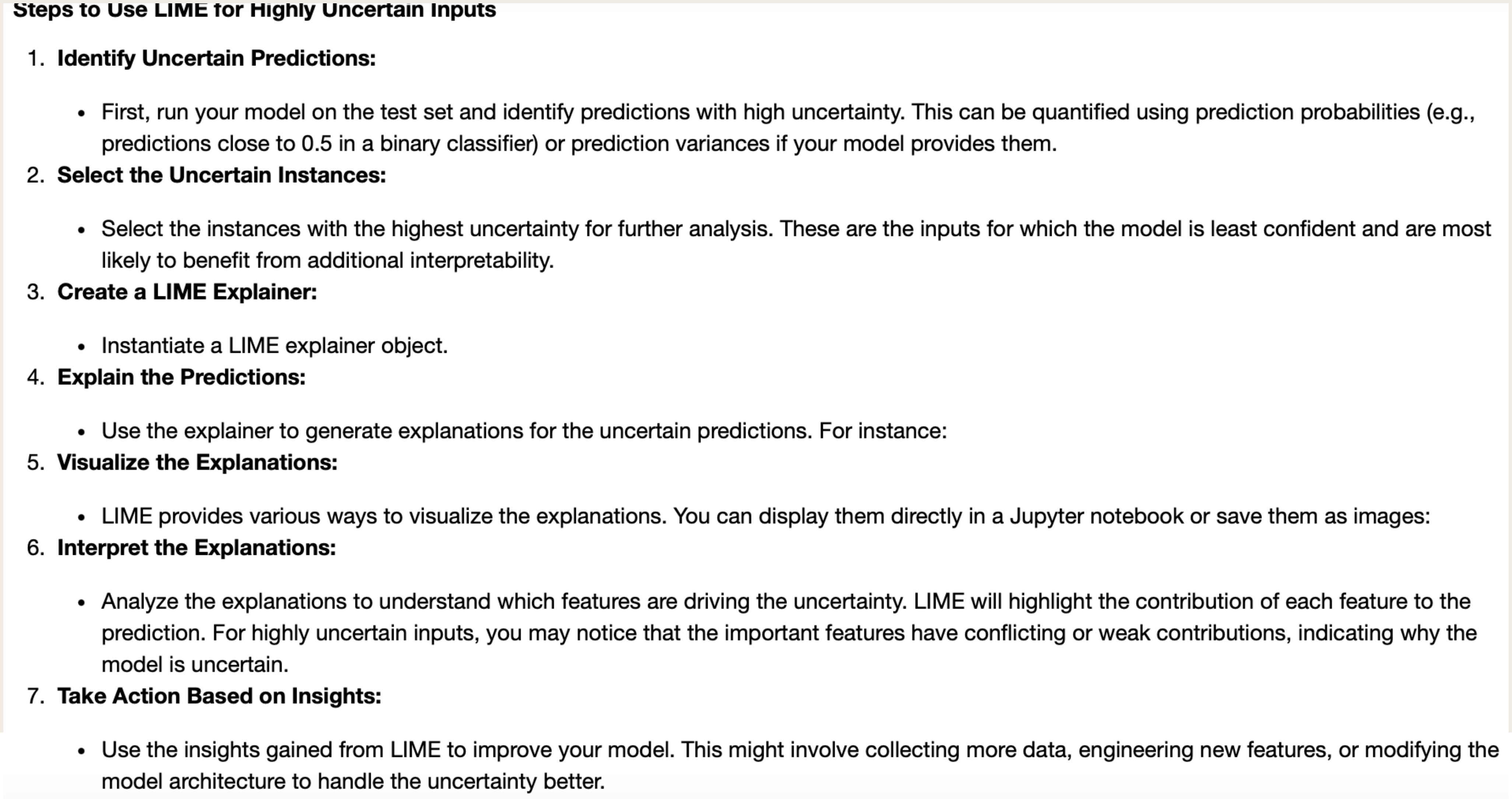
Specific Feature Contributions

- **Sepal Length (cm) = 5.8**: This feature has a significant positive contribution (red bar) of approximately 0.29 to the prediction, increasing the model's output towards the predicted class.
- **Sepal Width (cm) = 2.8**: This feature also has a positive contribution (red bar) but to a lesser extent.
- **Petal Width (cm) = 2.4**: This feature has a negative contribution (blue bar), decreasing the model's output.
- **Petal Length (cm) = 5.1**: This feature has a minor negative contribution (blue bar).

Highly Uncertain Instances with XAI

- When dealing with machine learning models, certain inputs can lead to highly uncertain predictions.
- This uncertainty can stem from various factors such as noisy data, complex decision boundaries, or insufficient training examples in specific regions of the feature space.
- Local Interpretable Model-agnostic Explanations (LIME) is a useful tool to understand and interpret these uncertain predictions.

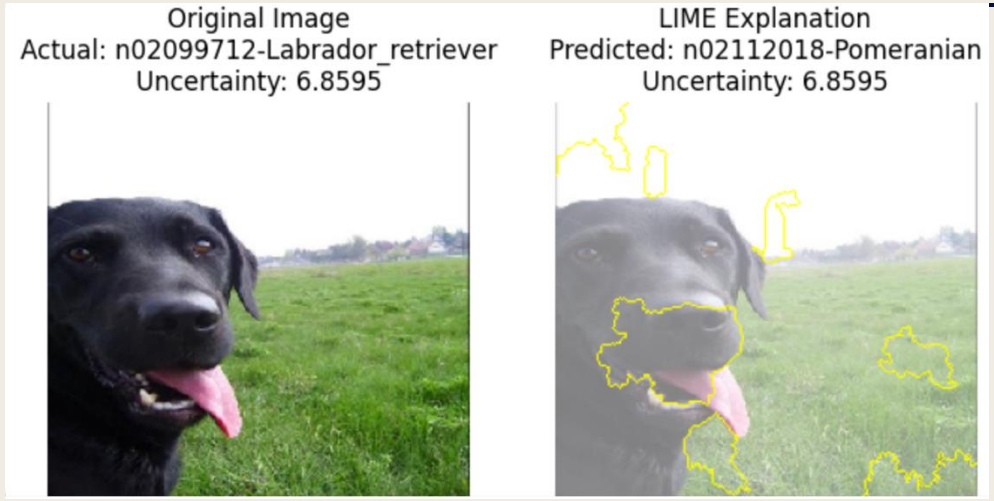
Highly Uncertain Instances with XAI

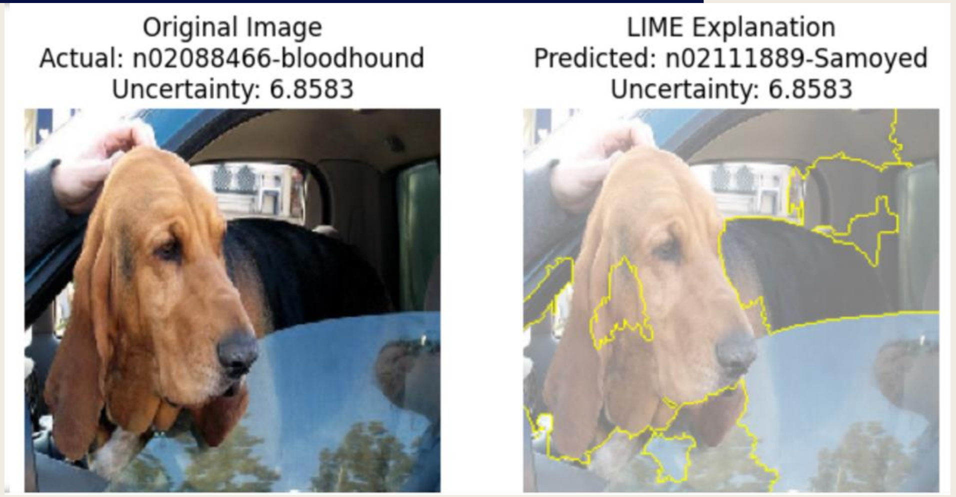


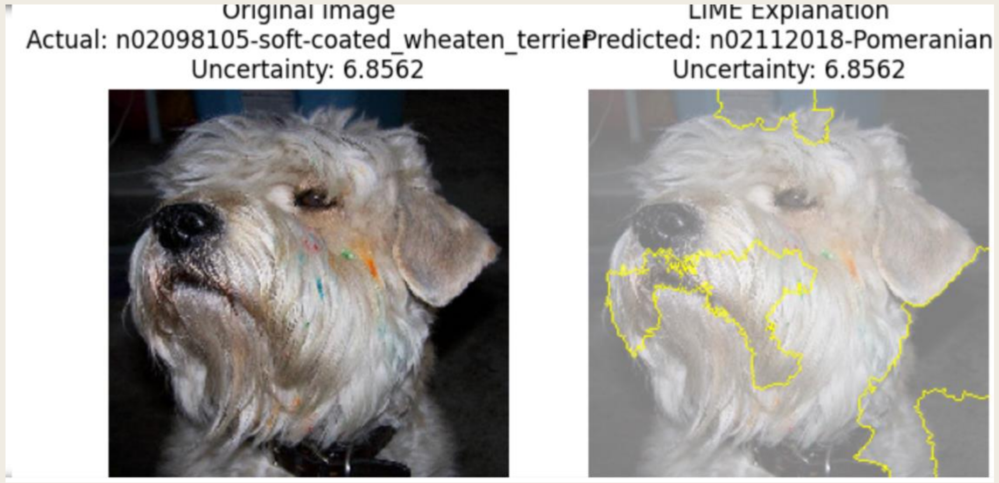
Steps to Use LIME for Highly Uncertain Inputs

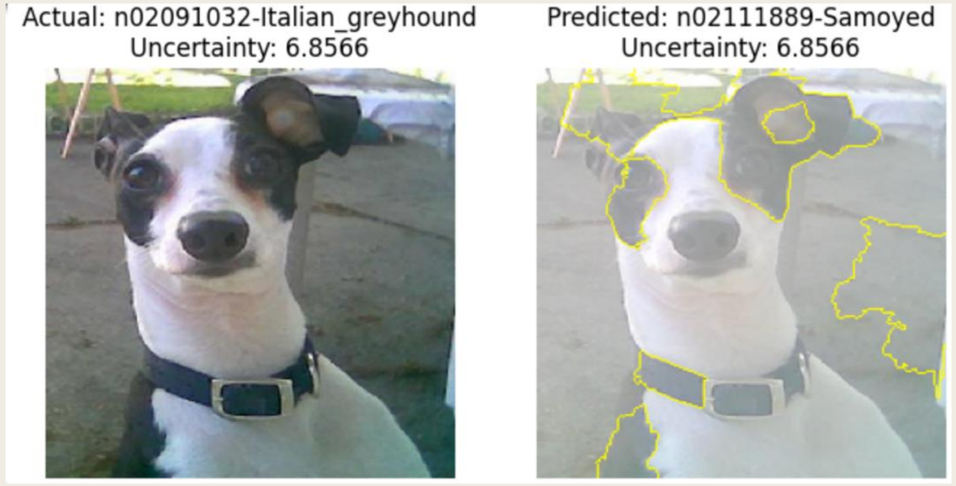
1. **Identify Uncertain Predictions:**
   - First, run your model on the test set and identify predictions with high uncertainty. This can be quantified using prediction probabilities (e.g., predictions close to 0.5 in a binary classifier) or prediction variances if your model provides them.
2. **Select the Uncertain Instances:**
   - Select the instances with the highest uncertainty for further analysis. These are the inputs for which the model is least confident and are most likely to benefit from additional interpretability.
3. **Create a LIME Explainer:**
   - Instantiate a LIME explainer object.
4. **Explain the Predictions:**
   - Use the explainer to generate explanations for the uncertain predictions. For instance:
5. **Visualize the Explanations:**
   - LIME provides various ways to visualize the explanations. You can display them directly in a Jupyter notebook or save them as images:
6. **Interpret the Explanations:**
   - Analyze the explanations to understand which features are driving the uncertainty. LIME will highlight the contribution of each feature to the prediction. For highly uncertain inputs, you may notice that the important features have conflicting or weak contributions, indicating why the model is uncertain.
7. **Take Action Based on Insights:**
   - Use the insights gained from LIME to improve your model. This might involve collecting more data, engineering new features, or modifying the model architecture to handle the uncertainty better.

Highly Uncertain Instances with LIME

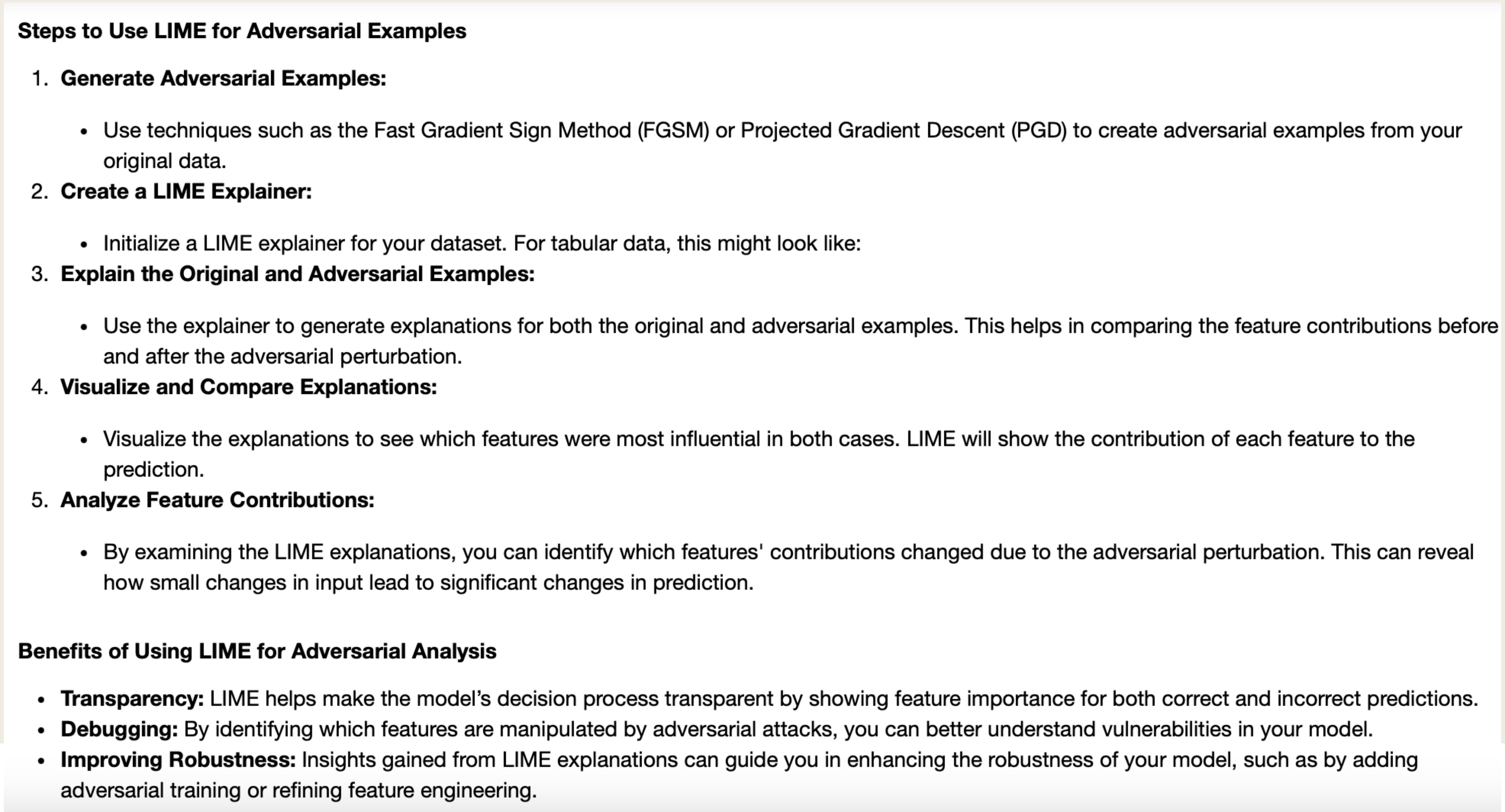





Original Image
Actual: n02098105-soft-coated_wheaten_terrie
Uncertainty: 6.8562
LIME Explanation
Predicted: n02112018-Pomeranian
Uncertainty: 6.8562


Actual: n02091032-Italian_greyhound
Uncertainty: 6.8566
Predicted: n02111889-Samoyed
Uncertainty: 6.8566

Adversarial Examples with XAI



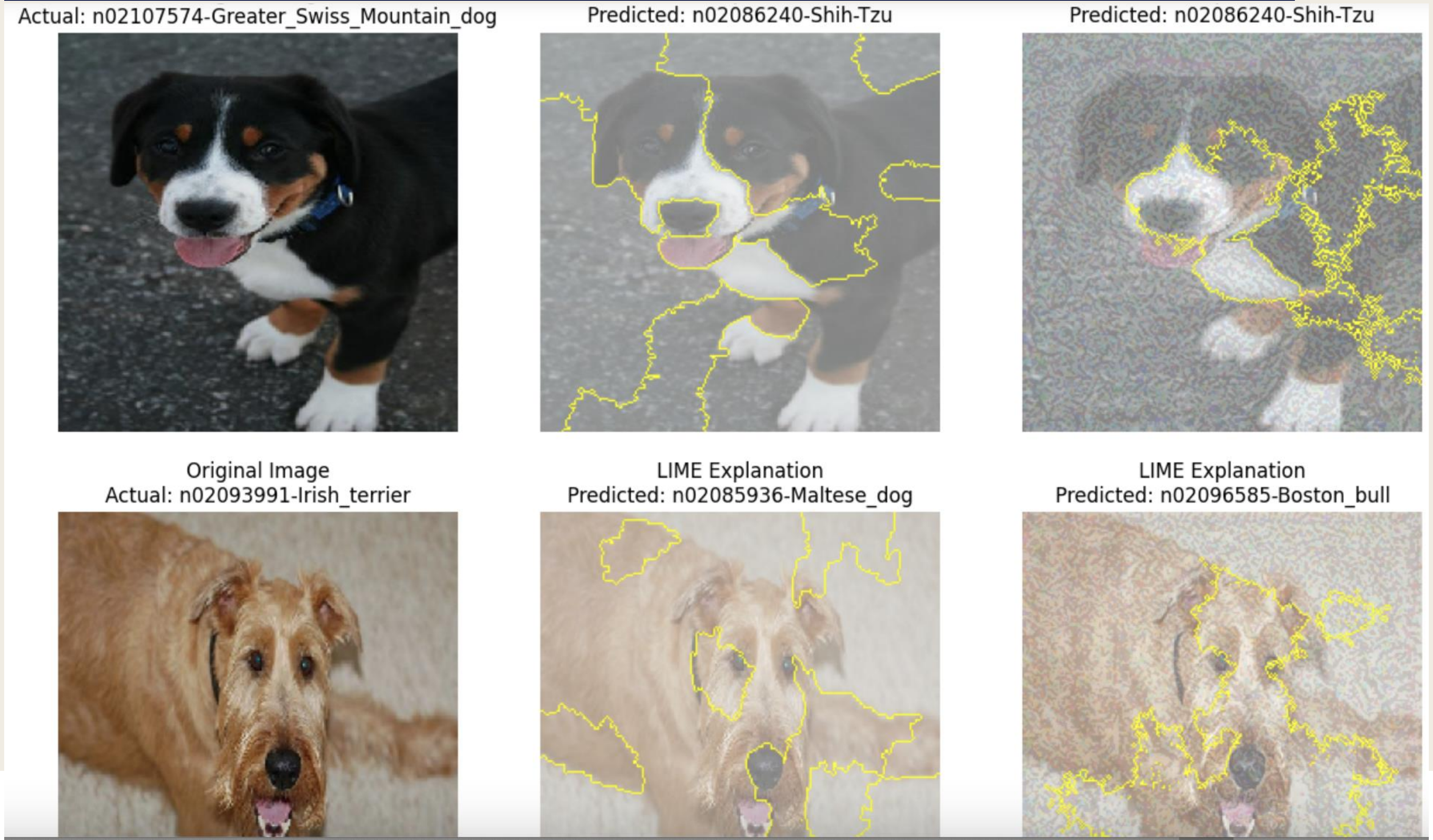
Steps to Use LIME for Adversarial Examples

1. **Generate Adversarial Examples:**
   - Use techniques such as the Fast Gradient Sign Method (FGSM) or Projected Gradient Descent (PGD) to create adversarial examples from your original data.
2. **Create a LIME Explainer:**
   - Initialize a LIME explainer for your dataset. For tabular data, this might look like:
3. **Explain the Original and Adversarial Examples:**
   - Use the explainer to generate explanations for both the original and adversarial examples. This helps in comparing the feature contributions before and after the adversarial perturbation.
4. **Visualize and Compare Explanations:**
   - Visualize the explanations to see which features were most influential in both cases. LIME will show the contribution of each feature to the prediction.
5. **Analyze Feature Contributions:**
   - By examining the LIME explanations, you can identify which features' contributions changed due to the adversarial perturbation. This can reveal how small changes in input lead to significant changes in prediction.

Benefits of Using LIME for Adversarial Analysis

- **Transparency:** LIME helps make the model's decision process transparent by showing feature importance for both correct and incorrect predictions.
- **Debugging:** By identifying which features are manipulated by adversarial attacks, you can better understand vulnerabilities in your model.
- **Improving Robustness:** Insights gained from LIME explanations can guide you in enhancing the robustness of your model, such as by adding adversarial training or refining feature engineering.

Adversarial Examples with XAI



Actual: n02107574-Greater_Swiss_Mountain_dog
Predicted: n02086240-Shih-Tzu
Predicted: n02086240-Shih-Tzu
Original Image
Actual: n02093991-Irish_terrier
LIME Explanation
Predicted: n02085936-Maltese_dog
LIME Explanation
Predicted: n02096585-Boston_bull