Machine Learning Model Explainability and the Trelawney Library

Brian Kreeger

July 9, 2020

About Me

- Data Science and Engineering Lead at Spok, Inc
- Rehabilitated SAS user
- Organizer of TCRUG and PyMNtos
- Father of 2 humans, a cat and a Great Dane

pymntos Twin Cities Python User Group

R Twin Cities R User Group

Agenda

- Model Explainability: Why is it important?
- Trelawney: an attempt to unify different existing APIs of current explainability
- Current tools: LIME and SHAP

Model Explainability

SR 11-7, Federal Reserve Guidance on Model Risk Management

“Banking organizations should be attentive to the possible adverse consequences (including financial loss) of decisions based on models that are incorrect or misused, and should address those consequences through active model risk management.”

FUTURE of AI Act (December 2, 2017)

Fundamentally Understanding The Usability and Realistic Evolution of Artificial Intelligence Act of 2017

European General Data Protection Regulation

“data subject shall have the right not to be subject to a decision based solely on automated processing” – Article 22

trelawney

Why Trelawney

Great character in Harry Potter series

Tries to provide two kinds of explanations (when possible):

- **global explanation** of the model that highlights the most important features the model uses to make its predictions globally (SHAP)
- **local explanation** of the model that will try to shed light on why a specific model made a specific prediction (LIME and SHAP)

Trelawnaey offers a unified API, representation and vocabulary for all explanation methods

Current Tools: LIME and SHAP

- LIME: Local Interpretable Model-agnostic Explanations

  Robeiro, Singh, Guestrin: Why Should I Trust You? Explaining the Predictions of Any Classifier (2016)

- SHAP: SHapely Additive exPlanation

  Lundberg, Erion, Lee: Consistent Individualized Feature Attribution for Tree Ensembles (2018)

Global Explainability

Trelawney provides two types of global explainers: SHAP and Surrogate explainer, which uses an interpretable model to mimic the outputs of our more complex model

Global Explainability: SHAP Explainer

In [32]:

```
from trelawney.shap_explainer import ShapExplainer

explainer = ShapExplainer()
explainer.fit(model, x_train, y_train)
```

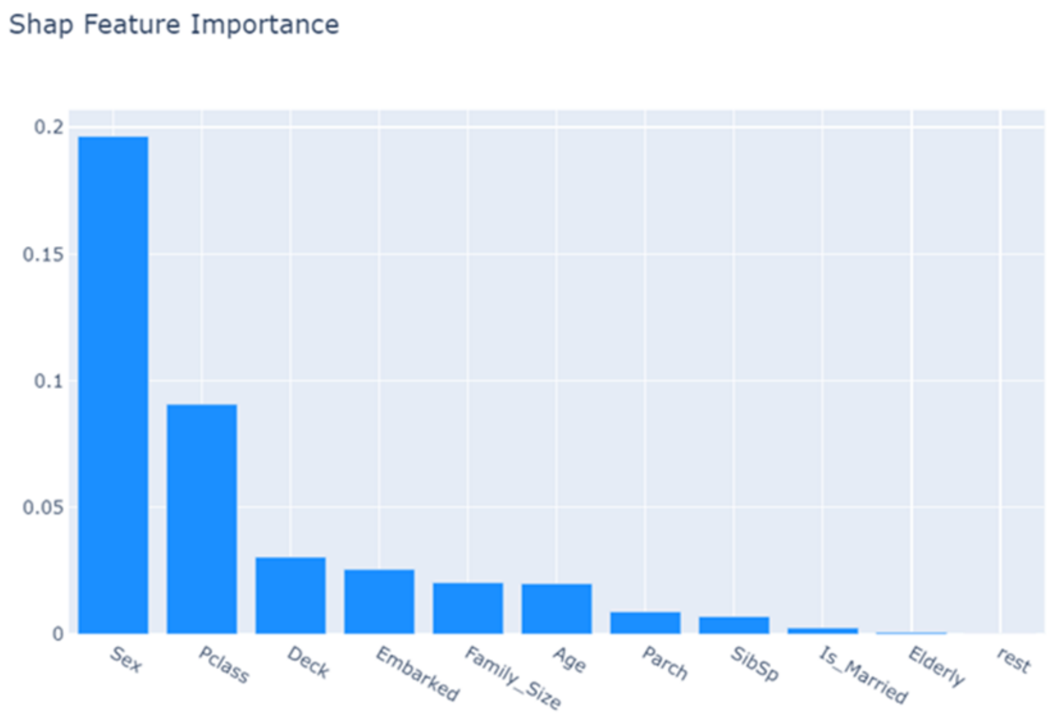
```
Using TensorFlow backend.
```

Out[32]:

```
<trelawney.shap_explainer.ShapExplainer at 0x7fb955b8b550>
```

In [33]:

```
feature_importance_graph = explainer.graph_feature_importance(x_val)
feature_importance_graph.update_layout(title='Shap Feature Importance')
feature_importance_graph.show()
```



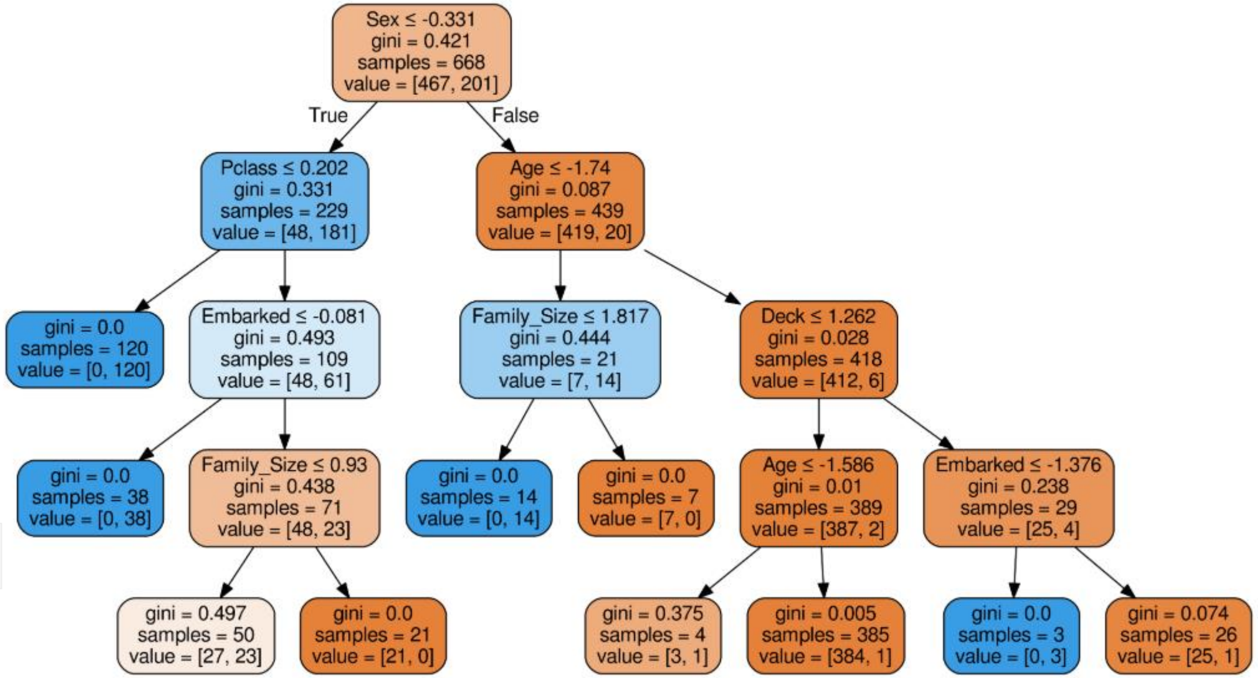
Global Explainability: Surrogate Explainer

In [34]:

```
from trelawney.surrogate_explainer import SurrogateExplainer
from sklearn.tree import DecisionTreeClassifier

explainer = SurrogateExplainer(DecisionTreeClassifier(max_depth=4))
explainer.fit(model, x_train, y_train)
```

Out[35]:



In [36]:

```
explainer.adequation_score()
```

Out[36]:

```
0.9610778443113772
```

Local Explainability

Trelawney tries to create local explainability around a specific predictions using both LIME and SHAP

LIME: Most Probable

In [39]:

```
y_pred = pd.DataFrame(model.predict_proba(x_val)[:, 1], index=x_val.index)
```

In [40]:

```
most_probable = y_pred.idxmax()
biggest_false_positive = (y_pred - y_val).idxmax()
biggest_false_negative = (y_pred - y_val).idxmin()
```

In [41]:

```
from trelawney.lime_explainer import LimeExplainer
```

In [42]:

```
explainer = LimeExplainer()
explainer.fit(model, x_train, y_train)
```

LIME: Most Probable

In [43]:

```
x_val.loc[most_probable, :]
```

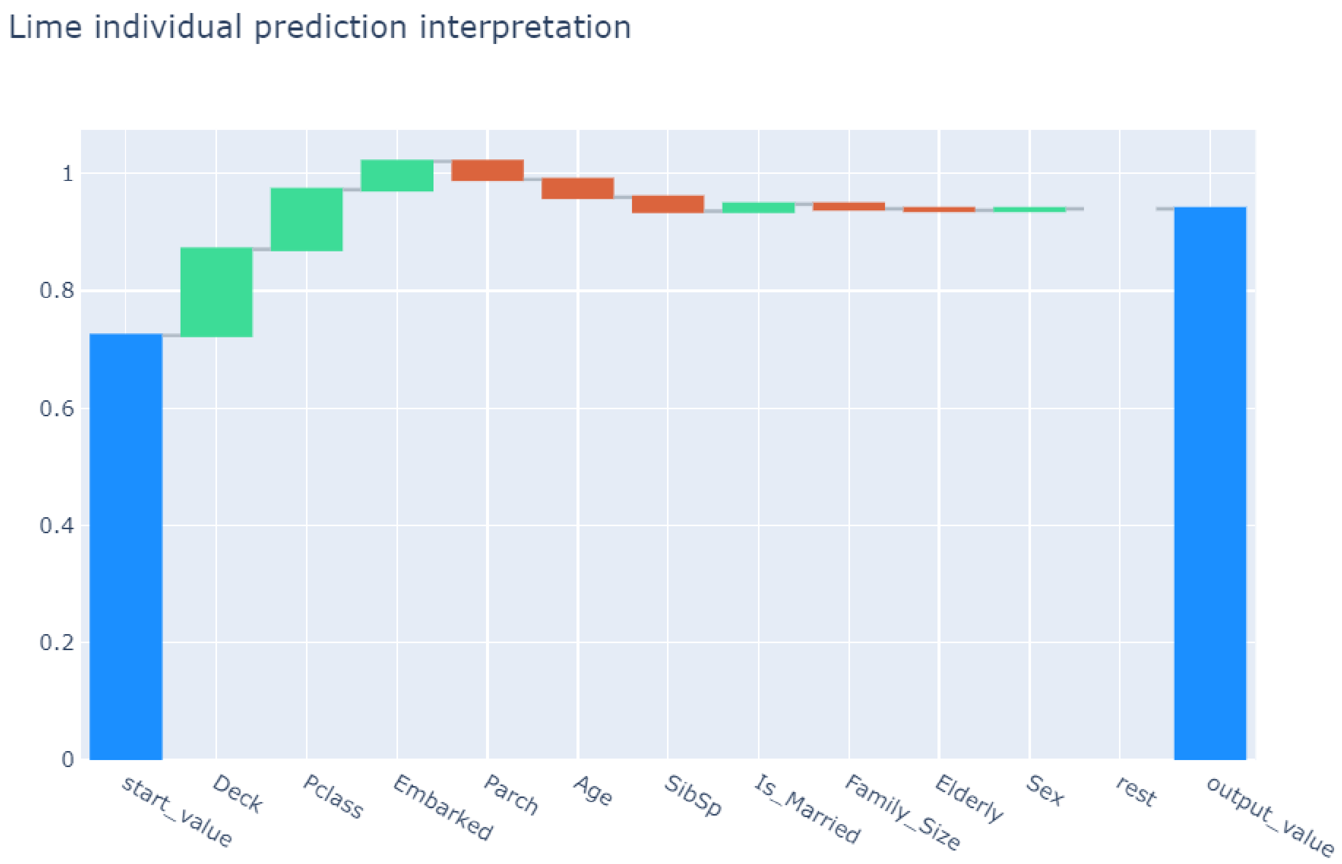
Out[43]:

	Pclass	Sex	Age	SibSp	Parch	Embarked	Deck	Is_Married	Family_Size	Elderly
215	-1.421307	-1.274178	0.075137	0.712002	-0.488106	-1.735018	1.961521	1.747726	0.119911	-0.3051

In [44]:

```
lime_explanation_graph = explainer.graph_local_explanation(x_val.loc[most_probable, :])
lime_explanation_graph.update_layout(title='Lime individual prediction interpretation')
lime_explanation_graph.show()
```

LIME: Most Probable



LIME: Biggest False Positive

In [45]:

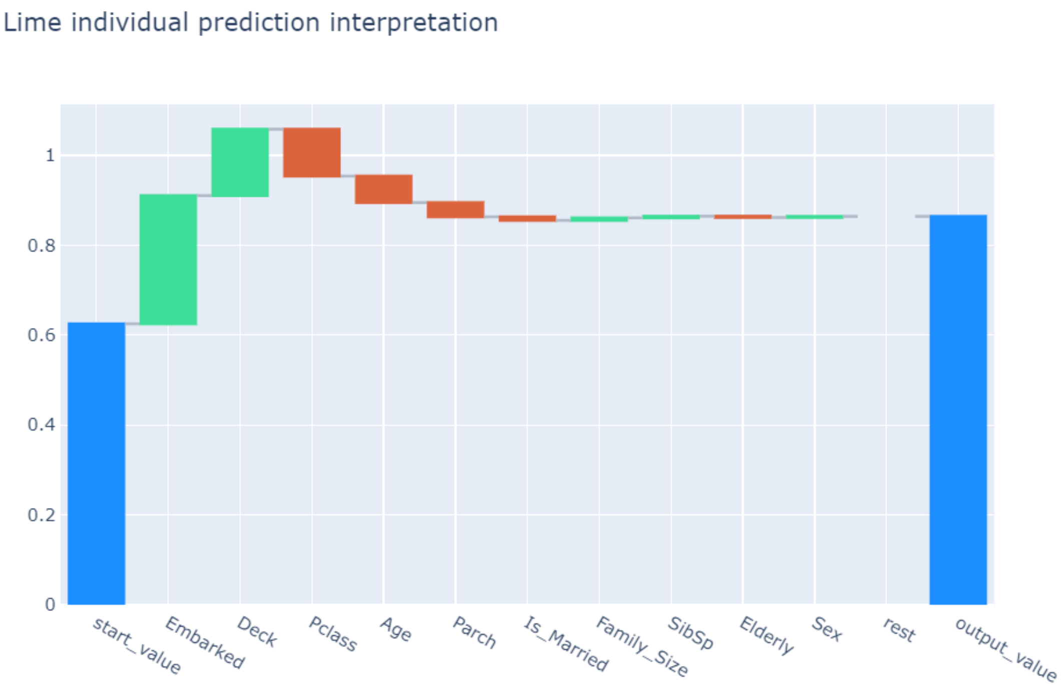
```
x.loc[biggest_false_positive, :]
```

Out[45]:

	Pclass	Sex	Age	SibSp	Parch	Embarked	Deck	Is_Married	Family_Size	Elderly
772	2	0	57.0	0	0	2	4	0	0	1

In [46]:

```
lime_explanation_graph = explainer.graph_local_explanation(x_val.loc[biggest_false_positive,
:])
lime_explanation_graph.update_layout(title='Lime individual prediction interpretation')
lime_explanation_graph.show()
```



SHAP

In [49]:

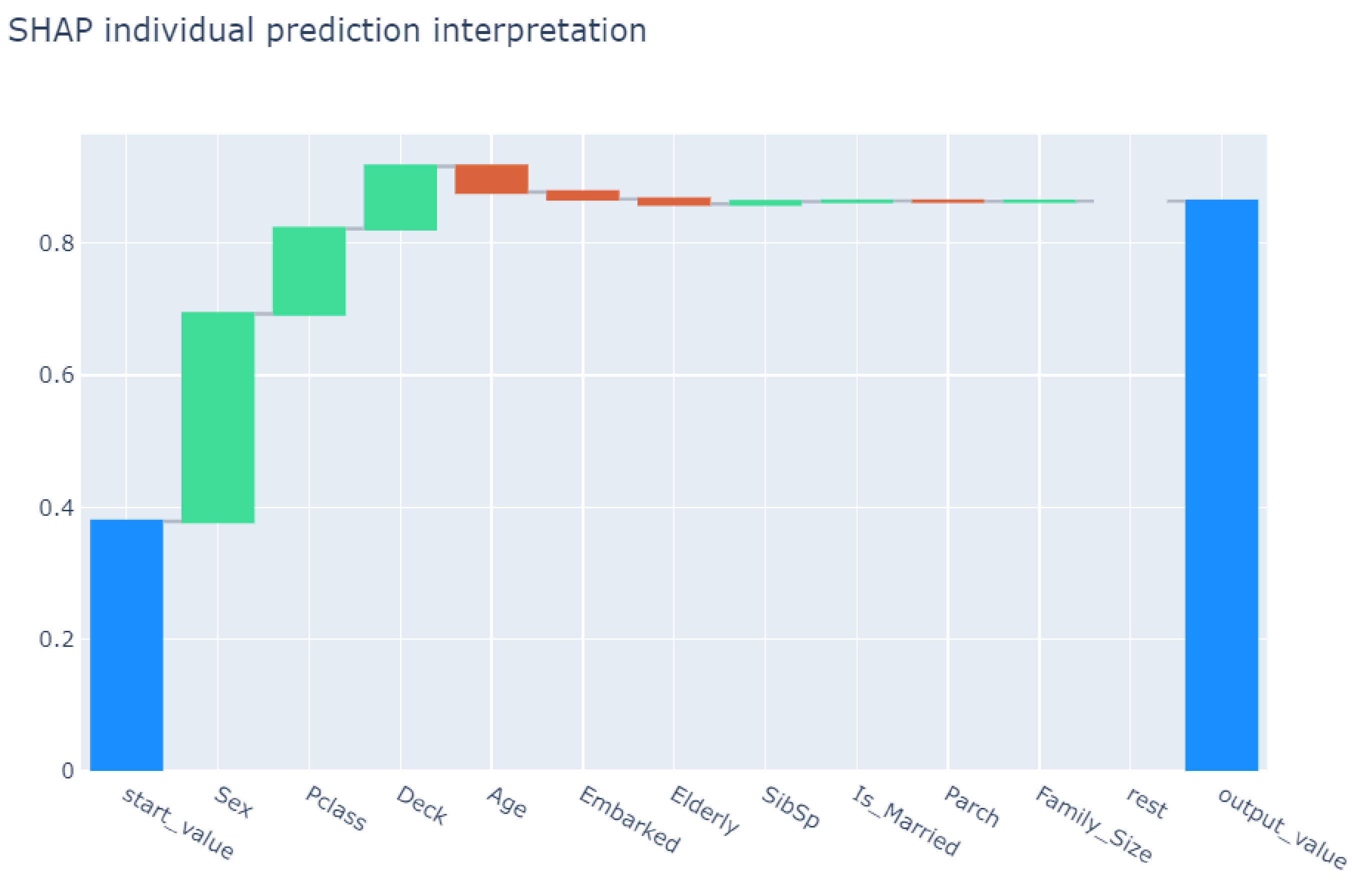
```
from trelawney.shap_explainer import ShapExplainer

explainer = ShapExplainer()
explainer.fit(model, x_train, y_train)
```

In [51]:

```
shap_explanation_graph = explainer.graph_local_explanation(x_val.loc[biggest_false_positive,
:])
shap_explanation_graph.update_layout(title='SHAP individual prediction interpretation')
shap_explanation_graph.show()
```

SHAP: Biggest False Negative



Vision

A simple(?) tool for multiple explainers to boost model transparency and code

Next Steps

Add support for regression (currently only supports classification models)

More model-focused methods

Better graphs (more interactivity?)

Links

Ribeiro, Singh and Guestrin LIME Paper

Lundberg, Erion and Lee SHAP Paper