

▼ Feature importance

This notebook is part of the following blogpost: <https://medium.com/bigdatarepublic/feature-importance-whats-in-a-name-79532e59eea3>

▼ Install dependencies

```
#Install non-standard packages (assuming jupyter notebook)
```

```
!pip install shap
```

```
!pip install lime
```

```
!pip install eli5
```

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-w> ▲
Collecting lime

Downloading lime-0.2.0.1.tar.gz (275 kB)

275.7/275.7 KB 4.4 MB/s eta 0:00:

Preparing metadata (setup.py) ... done

Requirement already satisfied: matplotlib in /usr/local/lib/python3.9/dist-packages

Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages

Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages

Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packages

Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.9/

Requirement already satisfied: scikit-image>=0.12 in /usr/local/lib/python3.9/

Requirement already satisfied: networkx>=2.2 in /usr/local/lib/python3.9/dist-

Requirement already satisfied: pillow!=7.1.0,!=7.1.1,!=8.3.0,>=6.1.0 in /usr/l

Requirement already satisfied: imageio>=2.4.1 in /usr/local/lib/python3.9/dist

Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dis

Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.9

Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.9/d

Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.

Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-

Requirement already satisfied: cycycler>=0.10 in /usr/local/lib/python3.9/dist-p

Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.9/di

Requirement already satisfied: importlib-resources>=3.2.0 in /usr/local/lib/py

Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.9/d

Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.9/di

Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.

Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.9/d

Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.9/dist-pa

Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packa

Building wheels for collected packages: lime

Building wheel for lime (setup.py) ... done

Created wheel for lime: filename=lime-0.2.0.1-py3-none-any.whl size=283859 s

Stored in directory: /root/.cache/pip/wheels/ed/d7/c9/5a0130d06d6310bc6cbe55

Successfully built lime

```

Successfully built lime
Installing collected packages: lime
Successfully installed lime-0.2.0.1
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-w
Collecting eli5
  Downloading eli5-0.13.0.tar.gz (216 kB)
    216.2/216.2 KB 4.6 MB/s eta 0:00:
  Preparing metadata (setup.py) ... done
Requirement already satisfied: attrs>17.1.0 in /usr/local/lib/python3.9/dist-p
Requirement already satisfied: Jinja2>=3.0.0 in /usr/local/lib/python3.9/dist-
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.9/dist-p
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: six in /usr/local/lib/python3.9/dist-packages (
Requirement already satisfied: scikit-learn>=0.20 in /usr/local/lib/python3.9/
Requirement already satisfied: graphviz in /usr/local/lib/python3.9/dist-packa
Requirement already satisfied: tabulate>=0.7.7 in /usr/local/lib/python3.9/dis
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dis
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-
Building wheels for collected packages: eli5
  Building wheel for eli5 (setup.py) ... done
  Created wheel for eli5: filename=eli5-0.13.0-py2.py3-none-any.whl size=10774
  Stored in directory: /root/.cache/pip/wheels/7b/26/a5/8460416695a992a2966b41
Successfully built eli5
Installing collected packages: eli5

```

▼ Import packages

```

#Imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import itertools

#scikit-learn package (https://pypi.org/project/scikit-learn)
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, auc, roc_auc
from sklearn.metrics import make_scorer

#eli5 package (https://eli5.readthedocs.io/en/latest)
import eli5
from eli5.sklearn import PermutationImportance

#lime package (https://github.com/marcotcr/lime)
import lime
import lime.lime_tabular

#shap package (https://github.com/slundberg/shap)
import shap

```

▼ Load data and train model

```
#Load Wisconsin diagnostic breast cancer data from UCI #

url_prefix = "https://archive.ics.uci.edu/ml/machine-learning-databases/"
#if url url does not work you can use this mirror instead:
#url_prefix = "http://mlr.cs.umass.edu/ml/machine-learning-databases/"

data_url = url_prefix + "breast-cancer-wisconsin/wdbc.data"

#Define column labels
id_status = ["ID", "diagnosis"]

#Mean, standard error and maximum ('worst') values are available for features
#computed on a collection of cells in a tissue
column_labels = itertools.product(["radius", "texture", "perimeter", "area",
                                   "smoothness", "compactness", "concavity",
                                   "concave_points", "symmetry", "fractal_dim"],
                                   ["mean", "std", "max"])

column_labels = id_status + [f"{t}_{f}" for f, t in column_labels]

#Read into pandas DataFrame
df = pd.read_csv(data_url, header=None, names = column_labels)

#Define feature set
X = df.drop('ID', axis=1).drop('diagnosis', axis=1)

#Define diagnosis as integer: malignant (1) or benign (0)
y = (df['diagnosis'] == "M")*1

#Split train and test set.
RANDOM_STATE = 123
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
                                                    stratify=y,
                                                    random_state=RANDOM_STATE)

#Train random forest classification model
model = RandomForestClassifier(max_depth=4, random_state=RANDOM_STATE)
model.fit(X_train, y_train)

# Diagnosis prediction
y_predict = model.predict(X_test)

# Probability of malignant tissue produced by the model
y_prob = [probs[1] for probs in model.predict_proba(X_test)]
```

▼ Evaluate model

```
#Accuracy on test set
print(f"Test accuracy: {accuracy_score(y_test, y_predict).round(2)}")

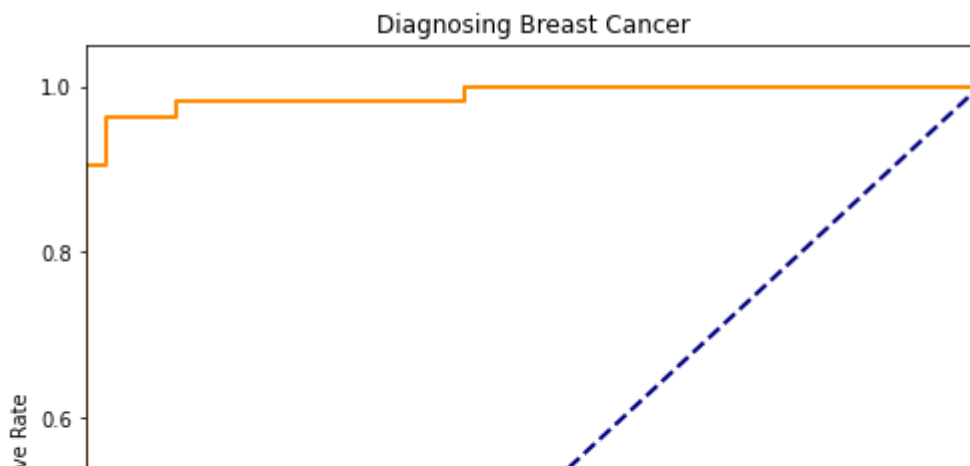
# Confusion matrix test set
pd.DataFrame(
    confusion_matrix(y_test, y_predict),
    columns=['Predicted Benign', 'Predicted Malignant'],
    index=['Benign', 'Malignant']
)

# Compute area under the curve
fpr, tpr, _ = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)

#Set default figure size
plt.rcParams['figure.figsize'] = (8,8)

# Plot ROC curve
plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange',
         lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title("Diagnosing Breast Cancer")
plt.legend(loc="lower right")
plt.show()
```

Test accuracy: 0.97



▼ Model-specific feature importance

```
# Feature importance dataframe
imp_df = pd.DataFrame({'feature': X_train.columns.values,
                       'importance': model.feature_importances_})

# Reorder by importance
ordered_df = imp_df.sort_values(by='importance')
imp_range=range(1,len(imp_df.index)+1)

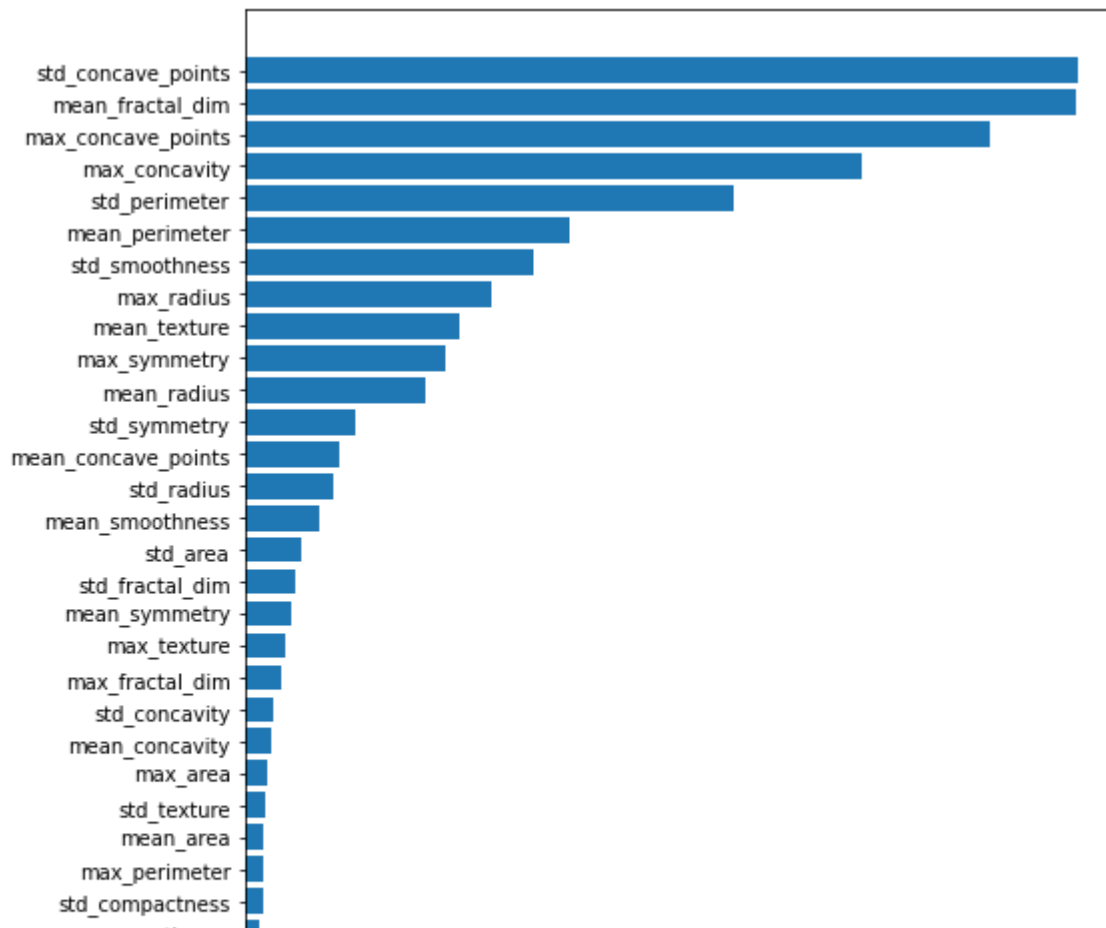
## Barplot with confidence intervals
height = ordered_df['importance']
bars = ordered_df['feature']
y_pos = np.arange(len(bars))

# Create horizontal bars
plt.barh(y_pos, height)

# Create names on the y-axis
plt.yticks(y_pos, bars)

plt.xlabel("Mean reduction in tree impurity in random forest")

plt.tight_layout()
# Show graphic
plt.show()
```



▼ Permutation feature importance

0.00 0.02 0.04 0.06 0.08 0.10 0.12 0.14

```
# Feature importance based on TRAINING set
```

```
perm_test = PermutationImportance(model, scoring=make_scorer(roc_auc_score),
                                   n_iter=50, random_state=RANDOM_STATE, cv="prefit")
```

```
# fit and see the permutation importances
```

```
perm_test.fit(X_train, y_train)
```

```
imp_df = eli5.explain_weights_df(perm_test)
```

```
label_df = pd.DataFrame({'feature': [ "x" + str(i) for i in range(len(X_test.columns))
```

```
imp_df = pd.merge(label_df, imp_df, on='feature', how='inner', validate="one_to_one")
```

```
# Reorder by importance
```

```
ordered_df = imp_df.sort_values(by='weight')
```

```
imp_range=range(1,len(imp_df.index)+1)
```

```
## Barplot with confidence intervals
```

```
height = ordered_df['weight']
```

```
bars = ordered_df['feature_name']
```

```
ci = 1.96 * ordered_df['std']
```

```

y_pos = np.arange(len(bars))

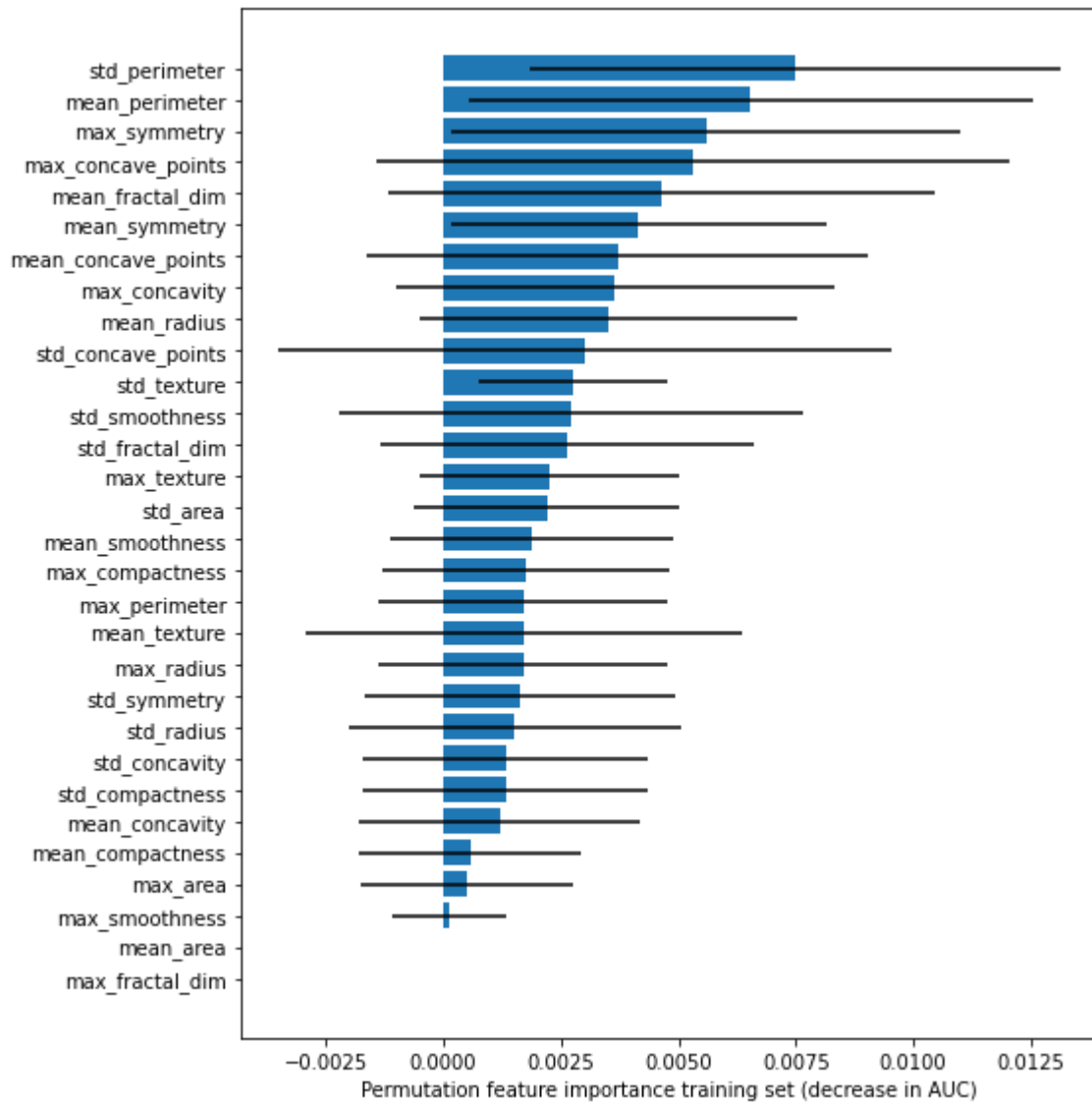
# Create horizontal bars
plt.barh(y_pos, height, xerr=ci)

# Create names on the y-axis
plt.yticks(y_pos, bars)

plt.xlabel("Permutation feature importance training set (decrease in AUC)")
plt.tight_layout()

# Show graphic
plt.show()

```

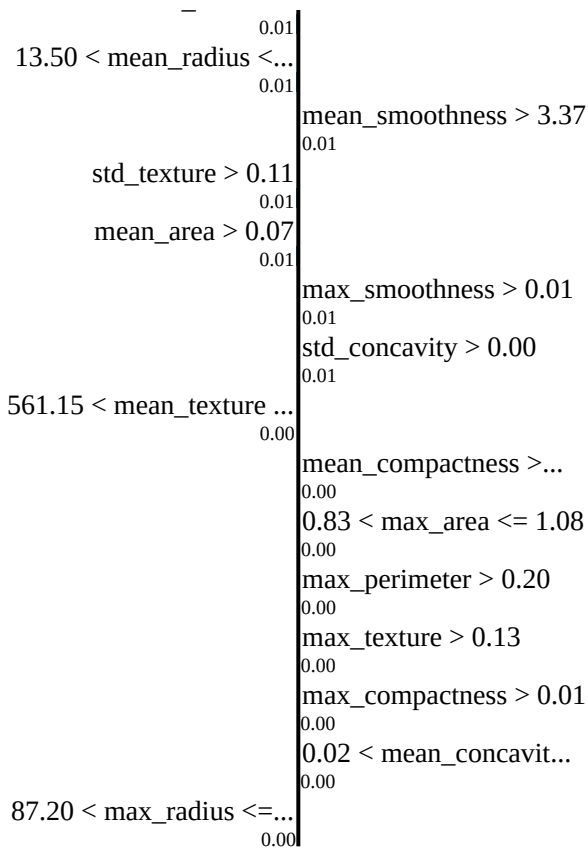


▼ LIME

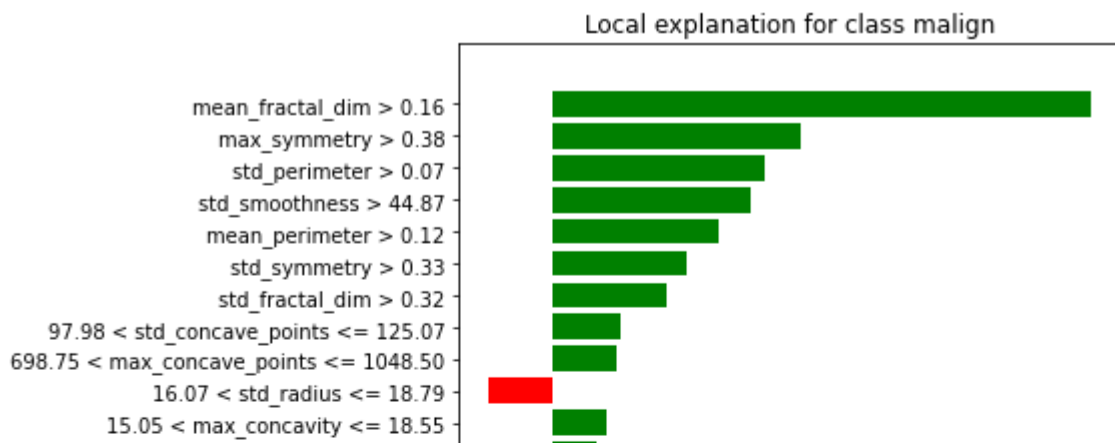
```
#Explain samples in test set
X_explain = X_test
explainer = lime.lime_tabular.LimeTabularExplainer(training_data=X_train.values,
                                                    feature_names=X_train.columns.values,
                                                    discretize_continuous=True,
                                                    class_names=["benign", "malign"],
                                                    mode="classification",
                                                    verbose=True,
                                                    random_state=RANDOM_STATE)

#Explaining first subject in test set using all 30 features
exp = explainer.explain_instance(X_explain.values[0,:],model.predict_proba,
                                num_features=30)

#Plot local explanation
plt = exp.as_pyplot_figure()
plt.tight_layout()
exp.show_in_notebook(show_table=True)
```

Feature	Value
mean_fractal_dim	0.22
max_symmetry	0.44
std_perimeter	0.12
std_smoothness	59.46
mean_perimeter	0.24
std_symmetry	0.45
std_fractal_dim	0.33
std_concave_points	119.80
max_concave_points	928.80
std_radius	17.27
max_concavity	17.73
std_area	0.66
std_compactness	0.05



explain the model's predictions on test set using SHAP values

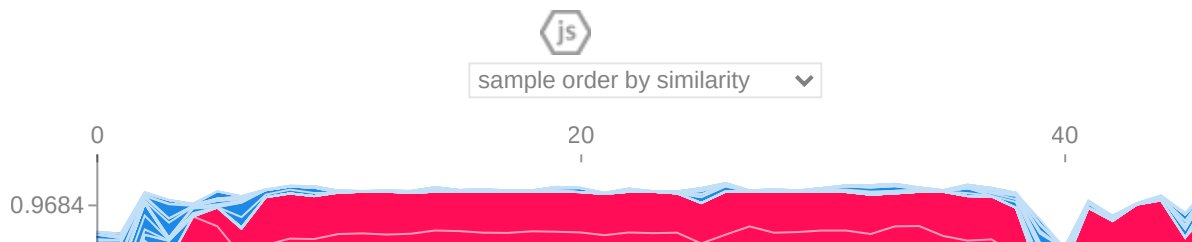
```
# same syntax works for xgboost, LightGBM, CatBoost, and some scikit-learn models
explainer = shap.TreeExplainer(model)

# shap_values consists of a list of two matrices of dimension samplesize x #features
# The first matrix uses average nr of benign samples as base value
# The second matrix which is used below uses average nr of malignant samples as base
shap_values = explainer.shap_values(X_explain)

# Interactive visualization of the explanation of the first subject
# in the test set (X_explain).
# It shows the relative contribution of features to get from the base value of malign
# samples(average value)
# to the output value (1 in case of malignant sample)
# the numbers at the bottom show the actual values for this sample.
shap.initjs() #initialize javascript in cell
shap.force_plot(explainer.expected_value[1], shap_values[1][0,:], X_explain.iloc[0,:])
```

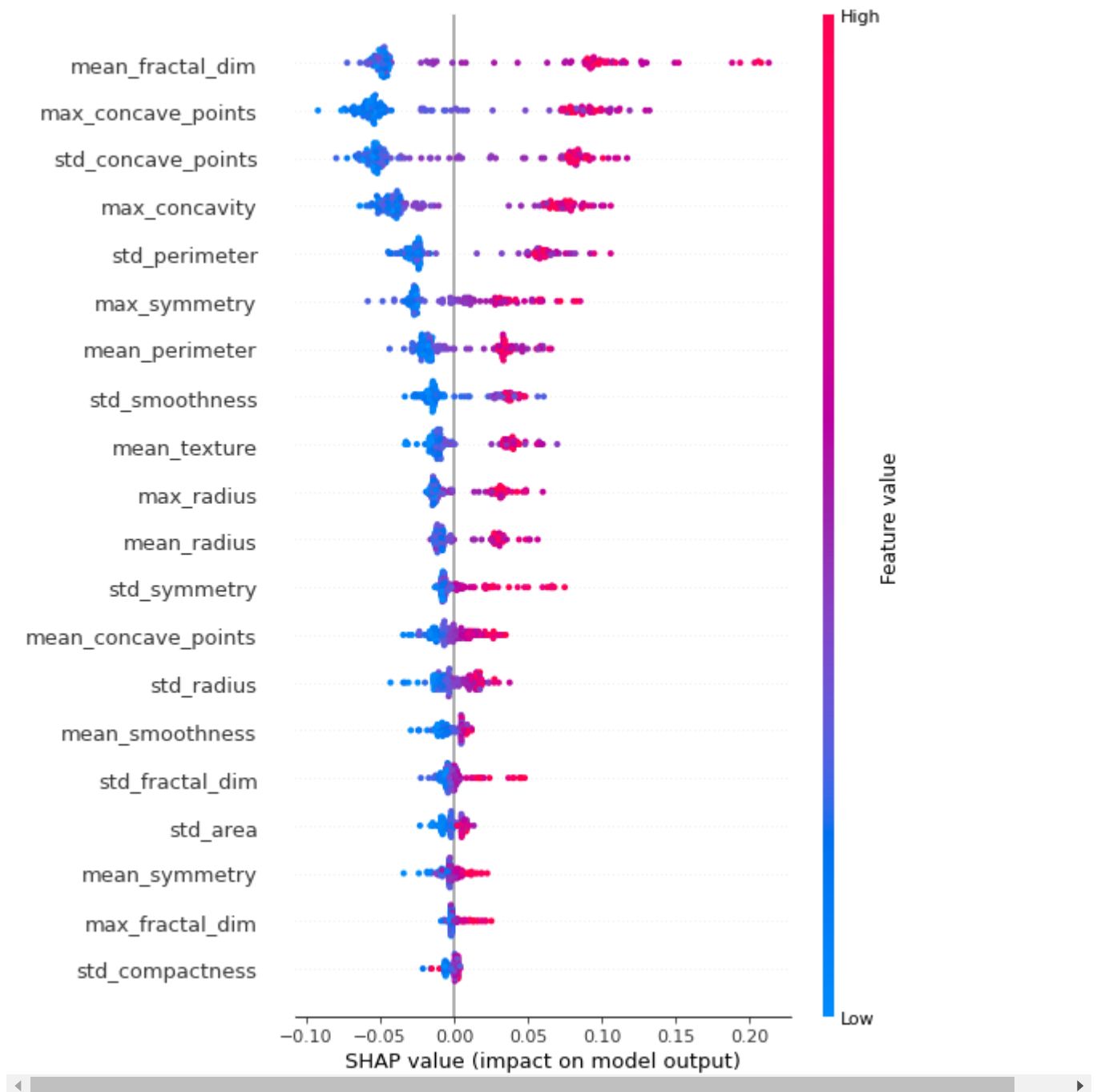


```
#Interactive visualization of all sample/feature Shapley values
#It is possible to show the relative contribution of individual features for all
# samples on the y-axis as well.
shap.initjs()
shap.force_plot(explainer.expected_value[1], shap_values[1], X_explain)
```



```
#A summary plot with the shapley value (feature importance)
shap.summary_plot(shap_values[1], X_explain)
```

No data for colormapping provided via 'c'. Parameters 'vmin', 'vmax' will be ignored



```
#Same as above, but with violin plots to better see the distribution of shapley value
shap.summary_plot(shap_values[1], X_explain, plot_type="violin")
```

➞ No data for colormapping provided via 'c'. Parameters 'vmin', 'vmax' will be ignored

