Sweetviz is designed to describe each variable and measure the degree of association among variables. This example uses the train.csv and test.csv files from the Titanic dataset.

```
!pip install sweetviz
import sweetviz
import pandas as pd
train = pd.read_csv("/content/train.csv")
test = pd.read_csv("/content/test.csv")
```

```
my_report = sweetviz.compare([train, "Train"], [test, "Test"], "Survived")
```

```
my_report.show_html("Report.html") # Not providing a filename will default to SWEETVIZ_REPORT.html
```

```
pip install lime
```

```
Requirement already satisfied: lime in /usr/local/lib/python3.6/dist-packages (0.2.0.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (from lime) (3.2.2)
Requirement already satisfied: scikit-image>=0.12 in /usr/local/lib/python3.6/dist-packages (from lime) (0.1
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages (from lime) (4.47.0)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.6/dist-packages (from lime) (0.2
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from lime) (1.18.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from lime) (1.4.1)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-packages (from matplotl
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib->lime
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib-
Requirement already satisfied: pillow>=4.3.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image>=0
Requirement already satisfied: PyWavelets>=0.4.0 in /usr/local/lib/python3.6/dist-packages (from scikit-imag
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image>=
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image>=0
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-learn>=0.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2.1
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.6/dist-packages (from networkx>=2.
```

Using LIME with a RandomForest Model

```
from sklearn.datasets import load_boston
import sklearn.ensemble
import numpy as np
from sklearn.model_selection import train_test_split
import lime
import lime.lime_tabular
```

```
boston = load_boston()
```

```
print(boston['DESCR'])
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the tar

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
```

```
rf = sklearn.ensemble.RandomForestRegressor(n_estimators=1000)
train, test, labels_train, labels_test = train_test_split(boston.data, boston.target, train_size=0.80)
rf.fit(train, labels_train)
```

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=1000, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

```
print('Random Forest MSError', np.mean((rf.predict(test) - labels_test) ** 2))
```

```
Random Forest MSError 7.070716154117681
```

```
print('MSError when predicting the mean', np.mean((labels_train.mean() - labels_test) ** 2))
```

```
MSError when predicting the mean 68.7759456493116
```

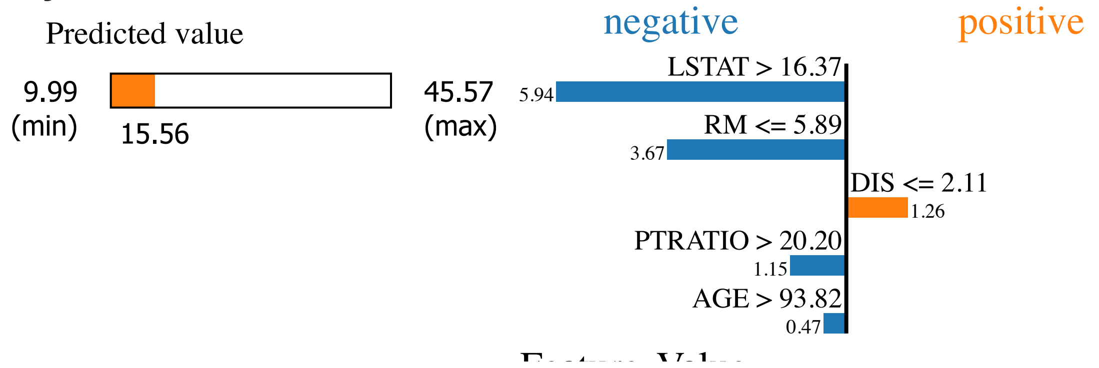
```
categorical_features = np.argwhere(
    np.array([len(set(boston.data[:,x]))
    for x in range(boston.data.shape[1])]) <= 10).flatten()
```

```
explainer = lime.lime_tabular.LimeTabularExplainer(train,
                                                   feature_names=boston.feature_names,
                                                   class_names=['price'],
                                                   categorical_features=categorical_features,
                                                   verbose=True, mode='regression')
```

```
i = 100

exp = explainer.explain_instance(test[i], rf.predict, num_features=5)
exp.show_in_notebook(show_table=True)
```

```
Intercept 26.111551565396354
Prediction_local [16.14733252]
Right: 15.562699999999997
```



Using LIME with a Linear Regression

```
import pandas as pd
from pandas import DataFrame
boston = load_boston()
boston.feature_names
dfx = pd.DataFrame(boston.data, columns = boston.feature_names)
dfy = pd.DataFrame(boston.target, columns = ['target'])
```

```
from sklearn.model_selection import train_test_split
X_train, y_train, X_test, y_test=train_test_split(boston.data, boston.target)
```

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(dfx,dfy)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
categorical_features = np.argwhere(
    np.array([len(set(boston.data[:,x]))
    for x in range(boston.data.shape[1])]) <= 10).flatten()
```

```
i = 100

exp = explainer.explain_instance(test[i], lr.predict, num_features=5)
exp.show_in_notebook(show_table=True)
```

```
Intercept 27.60191921042433
Prediction_local [22.19441596]
Right: 18.938685923806553
```