ML Visualization Cheat Sheet

Tools

With the help of following Python libraries, it makes it possible to understand ML data with statistics

YELLOWBRICK

- For Learning Curve:

```
from yellowbrick.model_selection import
learning_curve
```

- For Validation Curve:

```
from yellowbrick.model_selection import
validation_curve
```

- For Precision-Recall Curve:

```
from yellowbrick.classifier import
PrecisionRecallCurve
```

SKLEARN

- For ROC Curve:

```
from sklearn.metrics import roc_curve
```

- For Confusion Matrix:

```
from sklearn.metrics import
confusion_matrix
```

- For Precision-Recall Curve:

```
from sklearn.linear_model import
LinearRegression
```

SEABORN

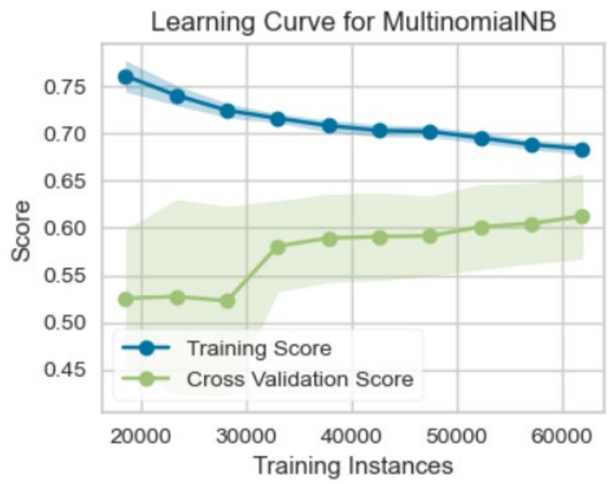
- For Confusion and Correlation Matrices:

```
import seaborn as sns
```

Classification

LEARNING CURVES

DO WE HAVE ENOUGH DATA?



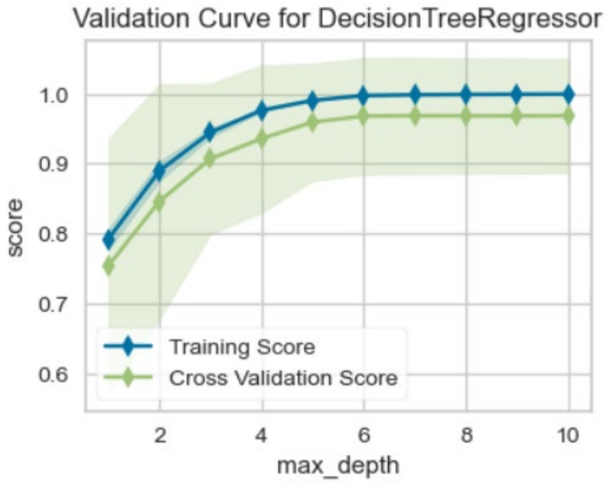
```
learning_curve(MultinomialNB(), X, y)
```

Where

- MultinomialNB() – classifier of your choice
- X, y – classification dataset

VALIDATON CURVES

WHAT ARE THE OPTIMAL HYPERPARAMETERS?



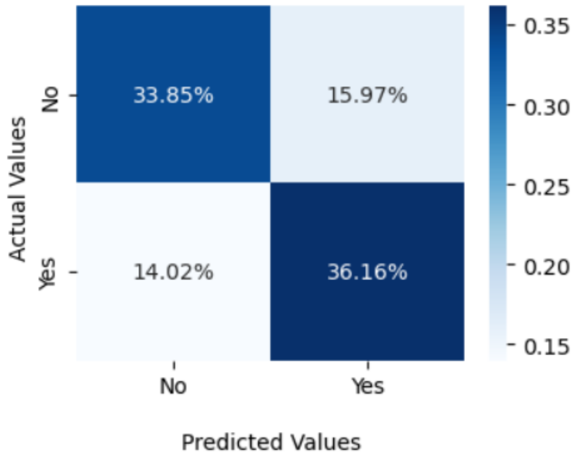
```
validation_curve(DecisionTreeRegressor(),
param_name="max_depth",
param_range=np.arange(1, 11))
```

Where

- DecisionTreeRegressor() – classifier of your choice
- param_name – name of the varied parameter

CONFUSION MATRIX

WHICH CLASSES ARE MIXED-UP?



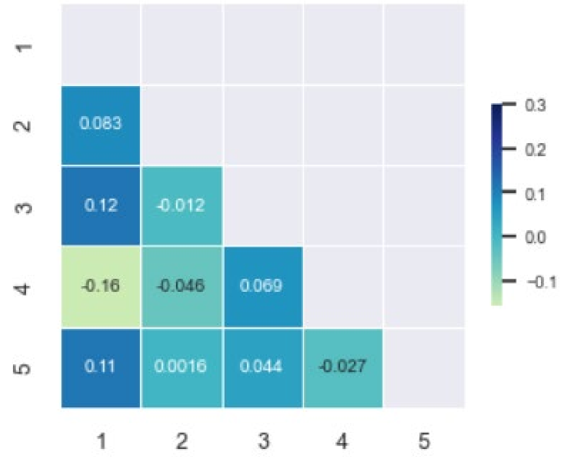
```
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm/np.sum(cm),fmt='.2%', annot=True,
cmap='Blues')
```

Where

- y_test, y_pred – test set, training set

CORRELATION MATRIX

HOW CORRELATED ARE PARAMETERS?



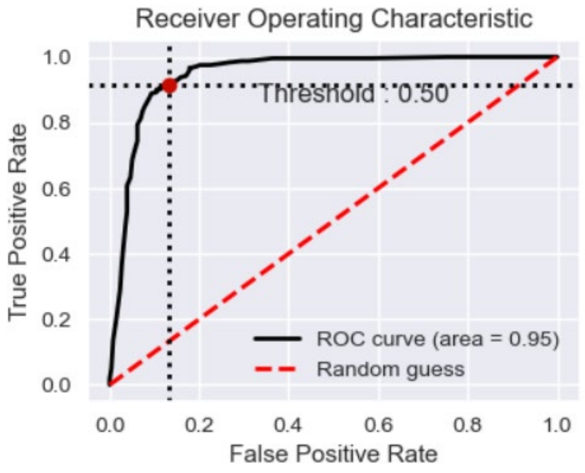
```
mask = np.triu(np.ones_like(dataset, dtype=bool))
sns.heatmap(dataset, mask=mask, cmap="YlGnBu",
annot=True, vmax=.3, center=0,
  square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

Where

- dataset – used dataset

ROC CURVE

IS CLASSIFIER A GOOD RANKER?



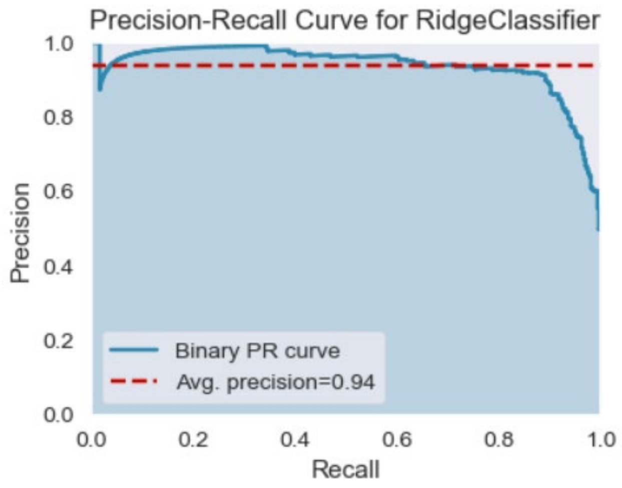
```
bc=BinaryClassification(y_test, y_pred, labels=[c1, c2])
bc.plot_roc_curve()
```

Where

- y_test, y_pred – test set, training set
- BinaryClassification() – classification of your choice

PRECISION-RECALL CURVE

WHAT IS THE PRECISION-RECALL TRADEOFF?



```
p = PrecisionRecallCurve(RidgeClassifier(random_state=0))
p.fit(X_train, y_train)
p.score(X_test, y_test)
```

Where

- X_train, X_test, y_test, y_pred – test sets, training sets

Updated: 2022-06

SHAP

- For Feature Importance Plot and SHAP summaries:

```
import xgboost
import shap
X, y = shap.dataset
model = xgboost.XGBRegressor().fit(X, y)
explainer = shap.Explainer(model)
```

Where

- model – an XGBoost model to train
- X, y – a dataset to train
- dataset – used dataset

YELLOWBRICK

- For Intercluster Distance Map:

```
from yellowbrick.cluster import
intercluster_distance
```

- For Residuals Plot:

```
from yellowbrick.regressor import
residuals_plot
```

- For Principal Component Plot:

```
from yellowbrick.features import PCA
```

PMDARIMA

- For Seasonality Decomposion:

```
from pmdarima import arima
from pmdarima import utils
```

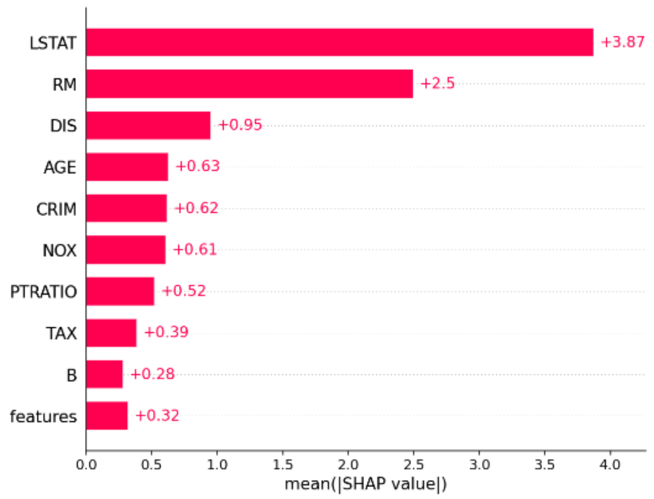
USEFUL LINKS

- SHAP
- YELLOWBRICK
- PMDARIMA
- SKLEARN
- SEABORN

Feature Analysis

FEATURE IMPORTANCE PLOT

MOST IMPORTANT PREDICTION FEATURES



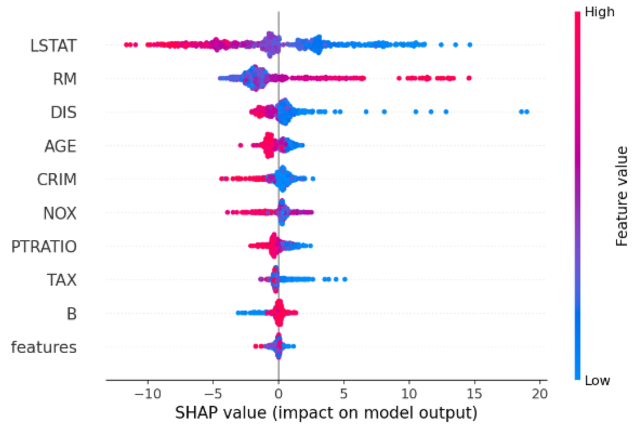
```
shap_values = explainer(X)
shap.plots.bar(shap_values)
```

Where

- shap_values – model's prediction explanation

SHAP SUMMARIES

FEATURE IMPORTANCES FOR EACH PREDICTION



```
shap_values = explainer(X)
shap.plots.bar(shap_values)
```

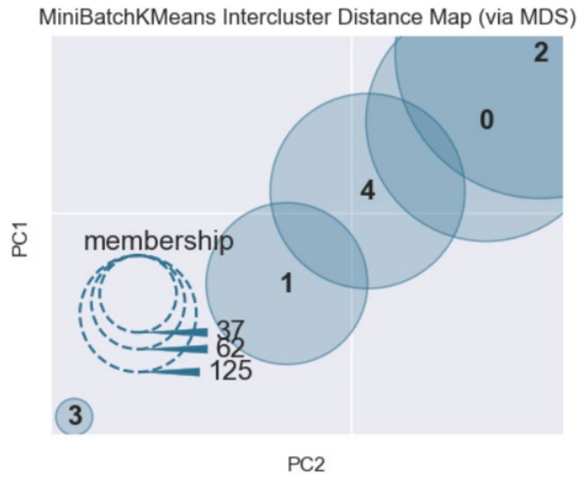
Where

- shap_values – model's prediction explanation

Clustering

INTERCLUSTER DISTANCE MAP

HOW DISTANT ARE THE CLUSTERS?



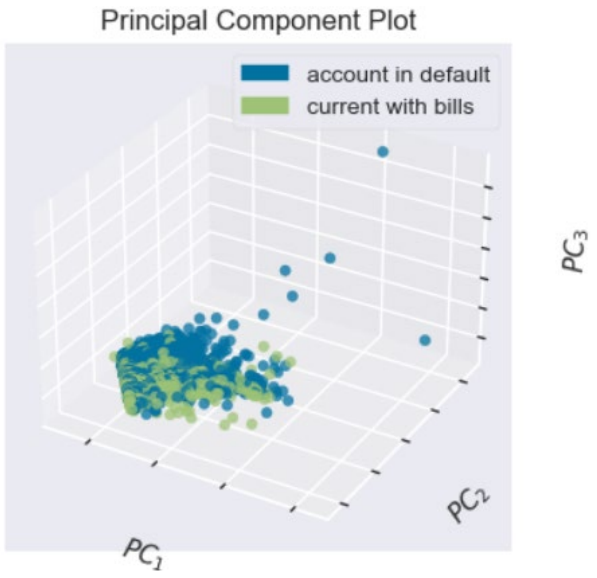
```
intercluster_distance(MiniBatchKMeans(5,
random_state=777), X)
```

Where

- MiniBatchKMeans – used algorithm

PRINCIPAL COMPONENT PLOT

3D-LOOK OF DATASET



```
p = PCA(scale=True, projection=3, classes=classes)
p.fit_transform(X, y)
```

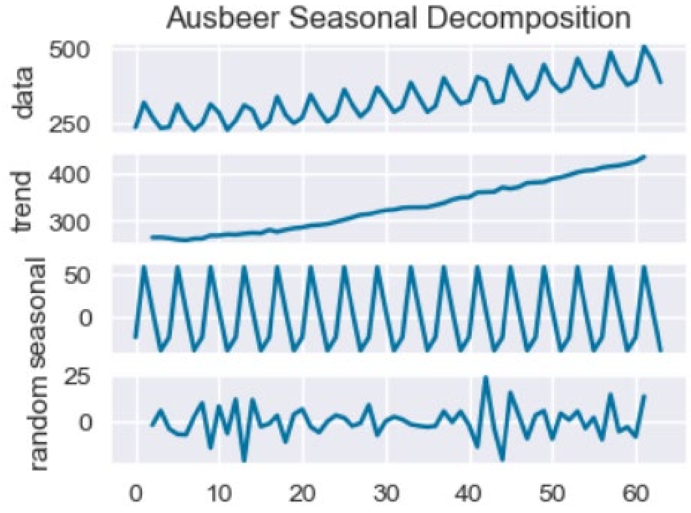
Where

- X, y – used dataset
- classes – class labels

Regression & Time Series

SEASONALITY DECOMPOSITION

SEASON, TRADE, NOISE IM TIME SERIES



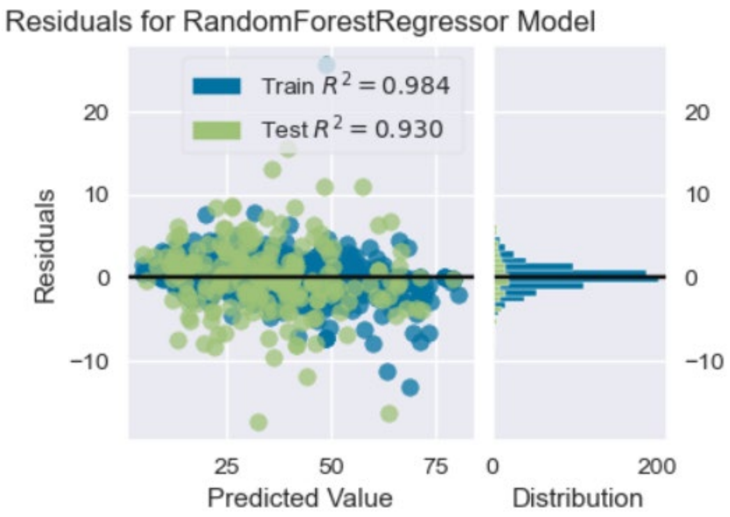
```
utils.decomposed_plot(decomposed,
figure_kwargs=figure_kwargs, show=False)
```

Where

- decomposed – tuple of y-axis variable datasets

RESIDUALS PLOT

ARE REGRESSION ERRORS NORMALLY DISTRIBUTED?



```
residuals_plot(RandomForestRegressor(), X_train,
y_train, X_test, y_test)
```

Where

- X_train, X_test, y_test, y_pred – test sets, training sets
- RandomForestRegressor() – estimator of your choice

Updated: 2022-06