Note

Click here to download the full example code

Quickstart - Model Evaluation Suite

The deepchecks model evaluation suite is relevant any time you wish to evaluate your model. For example:

- Thorough analysis of the model's performance before deploying it.
- Evaluation of a proposed model during the model selection and optimization stage.
- Checking the model's performance on a new batch of data (with or without comparison to previous data batches).

Here we'll build a regression model using the wine quality dataset (`deepchecks.tabular.datasets.regression.wine_quality`), to demonstrate how you can run the suite with only a few simple lines of code, and see which kind of insights it can find.

```
# Before we start, if you don't have deepchecks installed yet, run:
import sys
!{sys.executable} -m pip install deepchecks -U --quiet #--user

# or install using pip from your python environment
```

Prepare Data and Model

Load Data

```
from deepchecks.tabular.datasets.regression import wine_quality

data = wine_quality.load_data(data_format='Dataframe', as_train_test=False)
data.head(2)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.0	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.0	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5

Split Data and Train a Simple Model

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor

X_train, X_test, y_train, y_test = train_test_split(data.iloc[:, :-1], data['quality'], test_size=0.2)
gbr = GradientBoostingRegressor()
gbr.fit(X_train, y_train)
```

▾ GradientBoostingRegressor

GradientBoostingRegressor()

Run Deepchecks for Model Evaluation

Create a Dataset Object

Create a deepchecks Dataset, including the relevant metadata (label, date, index, etc.). Check out `deepchecks.tabular.Dataset` to see all the column types and attributes that can be declared.

```
from deepchecks.tabular import Dataset

# Categorical features can be heuristically inferred, however we
# recommend to state them explicitly to avoid misclassification.

# Metadata attributes are optional. Some checks will run only if specific attributes are declared.

train_ds = Dataset(X_train, label=y_train, cat_features=[])
test_ds = Dataset(X_test, label=y_test, cat_features=[])
```

Run the Deepchecks Suite

Validate your data with the **deepchecks.tabular.suites.model_evaluation** suite. It runs on two datasets and a model, so you can use it to compare the performance of the model between any two batches of data (e.g. train data, test data, a new batch of data that recently arrived)

Check out the “when should you use deepchecks guide” for some more info about the existing suites and when to use them.

```
from deepchecks.tabular.suites import model_evaluation

evaluation_suite = model_evaluation()
suite_result = evaluation_suite.run(train_ds, test_ds, gbr)
# Note: the result can be saved as html using suite_result.save_as_html()
# or exported to json using suite_result.to_json()
suite_result.show()
```

Out:

```
Model Evaluation Suite:
|            | 0/12 [Time: 00:00]
Model Evaluation Suite:
|#           | 1/12 [Time: 00:00, Check=Performance Report]
Model Evaluation Suite:
|#####       | 5/12 [Time: 00:00, Check=Simple Model Comparison]
Model Evaluation Suite:
|#####       | 5/12 [Time: 00:20, Check=Weak Segments Performance]
Model Evaluation Suite:
|######      | 6/12 [Time: 00:20, Check=Weak Segments Performance]
Model Evaluation Suite:
|#########   | 9/12 [Time: 00:20, Check=Regression Error Distribution]
Model Evaluation Suite:
|############| 12/12 [Time: 00:20, Check=Model Inference Time]

/Users/nadav/Desktop/GitRepos/deepchecks/deepchecks/core/serialization/suite_result/widget.py:154:
FutureWarning:

this method is deprecated in favour of `Styler.hide(axis='index')`
```

▾ **Model Evaluation Suite**

Model Evaluation Suite

The suite is composed of various checks such as: Weak Segments Performance, Confusion Matrix Report, Unused Features, etc...

Each check may contain conditions (which will result in pass / fail / warning / error , represented by ✓ / ✖ / ! / ?!) as well as other outputs such as plots or tables.

Suites, checks and conditions can all be modified. Read more about custom suites.

▸ **Didn't Pass**

▸ **Passed**

▸ **Other**

▸ **Didn't Run**

▾ **Model Evaluation Suite**

Model Evaluation Suite

The suite is composed of various checks such as: Weak Segments Performance, Confusion Matrix Report, Unused Features, etc...

Each check may contain conditions (which will result in pass / fail / warning / error , represented by ✓ / ✖ / ! / ?!) as well as other outputs such as plots or tables.

Suites, checks and conditions can all be modified. Read more about custom suites.

▸ **Didn't Pass**

▸ **Passed**

▸ **Other**

▸ **Didn't Run**

Analyzing the results

The result showcase a number of interesting insights, first lets inspect the “Didn’t Pass” section.

- /checks_gallery/tabular/model_evaluation/plot_performance_report check result implies that the model overfitted the training data.
- /checks_gallery/tabular/model_evaluation/plot_regression_systematic_error (test set) check result demonstrate the model small positive bias.
- /checks_gallery/tabular/model_evaluation/plot_weak_segments_performance (test set) check result visualize some specific sub-spaces on which the model performs poorly. Examples for those sub-spaces are wines with low total sulfur dioxide and wines with high alcohol percentage.

Next, let us examine the “Passed” section.

- /checks_gallery/tabular/model_evaluation/plot_simple_model_comparison check result states that the model performs better than naive baseline models, an opposite result could indicate a problem with the model or the data it was trained on.
- /checks_gallery/tabular/model_evaluation/plot_boosting_overfit check and the /checks_gallery/tabular/model_evaluation/plot_unused_features check results implies that the model has a well calibrating boosting stopping rule and that it make good use on the different data features.

Let us try and fix the overfitting issue found in the model.

Fix the Model and Re-run the Model Evaluation Suite

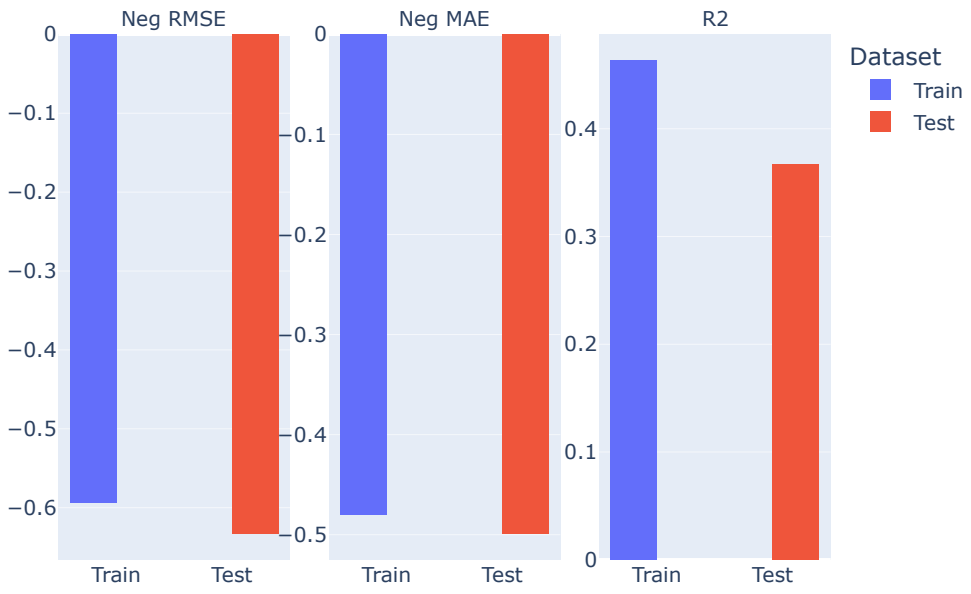
```
from deepchecks.tabular.checks import PerformanceReport

gbr = GradientBoostingRegressor(n_estimators=20)
gbr.fit(X_train, y_train)
PerformanceReport().run(train_ds, test_ds, gbr)
```

Performance Report

Summarize given scores on a dataset and model.

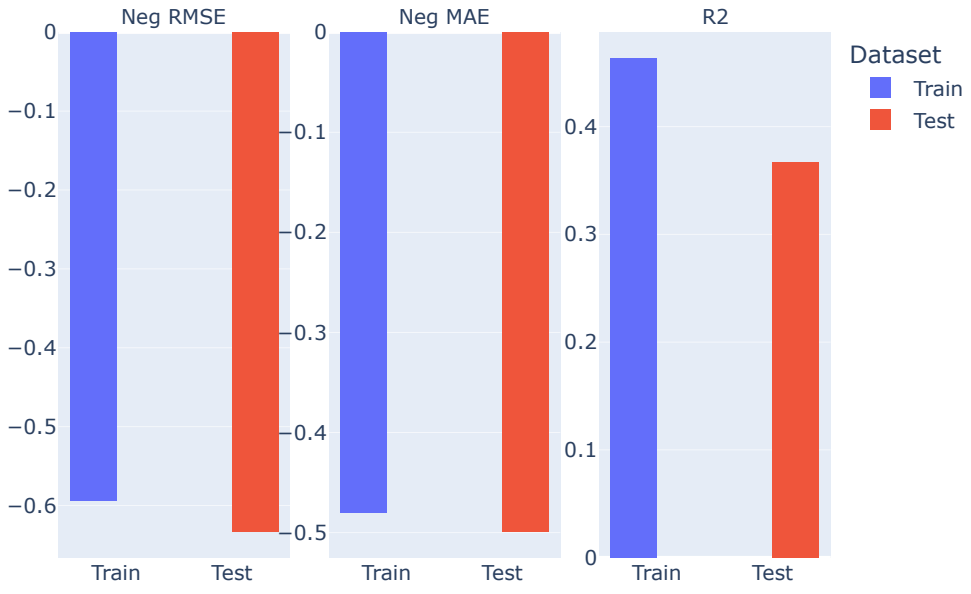
Additional Outputs



Performance Report

Summarize given scores on a dataset and model.

Additional Outputs



Search the docs ...

Quickstart - Model Evaluation Suite

We mitigated the overfitting to some extent. Additional model tuning to overcome the other issues discussed above however for now we will update and remove the relevant conditions from the suite.

Updating an Existing Suite

To create our own suite, we can start with an empty suite and add checks and condition to it (see /user-guide/general/customizations/examples/plot_create_a_custom_suite), or we can start with one of the default suites and update it as demonstrated in this section.

let's inspect our model evaluation suite's structure

```
evaluation_suite
```

Out:

```
Model Evaluation Suite: [
    0: PerformanceReport
            Conditions:
                    0: Train-Test scores relative degradation is less than 0.1
    1: RocReport(excluded_classes=[])
            Conditions:
                    0: AUC score for all the classes is greater than 0.7
    2: ConfusionMatrixReport
    3: TrainTestPredictionDrift
            Conditions:
                    0: categorical drift score < 0.15 and numerical drift score < 0.075
    4: SimpleModelComparison
            Conditions:
                    0: Model performance gain over simple model is greater than 10%
    5: WeakSegmentsPerformance(n_to_show=5)
            Conditions:
                    0: The relative performance of weakest segment is greater than 80% of average
model performance.
    6: CalibrationScore
    7: RegressionSystematicError
            Conditions:
                    0: Bias ratio is less than 0.01
    8: RegressionErrorDistribution
            Conditions:
```

Next, we will update the Performance Report condition and remove the Regression Systematic Error check:

```
evaluation_suite[0].clean_conditions()
evaluation_suite[0].add_condition_train_test_relative_degradation_less_than(0.3)
evaluation_suite = evaluation_suite.remove(7)
```

Re-run the suite using:

```
result = evaluation_suite.run(train_ds, test_ds, gbr)
result.passed(fail_if_warning=False)
```

Out:

```
Model Evaluation Suite:
|           | 0/11 [Time: 00:00]
Model Evaluation Suite:
|#          | 1/11 [Time: 00:00, Check=Performance Report]
Model Evaluation Suite:
|#####      | 5/11 [Time: 00:00, Check=Simple Model Comparison]
Model Evaluation Suite:
|########   | 8/11 [Time: 00:02, Check=Regression Error Distribution]
Model Evaluation Suite:
|########## | 10/11 [Time: 00:03, Check=Boosting Overfit]

True
```

For more info about working with conditions, see the detailed /user-guide/general/customizations/examples/plot_configure_check_conditions guide.

Total running time of the script: (0 minutes 29.016 seconds)

Download Python source code: plot_test.py

Download Jupyter notebook: plot_test.ipynb

Previous
Plots

Next
Example RST Notebook