

---

# **ebm2onnx Documentation**

***Release 3.1.1***

**Romain Picard**

**Mar 07, 2023**



**CONTENTS:**

|          |                                    |           |
|----------|------------------------------------|-----------|
| <b>1</b> | <b>Ebm2onnx</b>                    | <b>1</b>  |
| 1.1      | Features . . . . .                 | 1         |
| 1.2      | Get Started . . . . .              | 1         |
| 1.3      | Try it live . . . . .              | 2         |
| 1.4      | Supporting organizations . . . . . | 3         |
| <b>2</b> | <b>Installation</b>                | <b>5</b>  |
| 2.1      | Stable release . . . . .           | 5         |
| 2.2      | From sources . . . . .             | 5         |
| <b>3</b> | <b>Usage</b>                       | <b>7</b>  |
| <b>4</b> | <b>Reference</b>                   | <b>9</b>  |
| <b>5</b> | <b>Indices and tables</b>          | <b>11</b> |
|          | <b>Python Module Index</b>         | <b>13</b> |
|          | <b>Index</b>                       | <b>15</b> |



## EBM2ONNX

Ebm2onnx converts [EBM](#) models to ONNX. It allows to run an EBM model on any ONNX compliant runtime.

### 1.1 Features

- Binary classification
- Regression
- Continuous, nominal, and ordinal variables
- N-way interactions
- Multi-class classification (support is still experimental in EBM)
- Expose predictions probabilities
- Expose local explanations

The export of the models is tested against [ONNX Runtime](#).

### 1.2 Get Started

Train an EBM model:

```
# prepare dataset
df = pd.read_csv('titanic_train.csv')
df = df.dropna()

feature_columns = ['Age', 'Fare', 'Pclass', 'Embarked']
label_column = "Survived"
y = df[[label_column]]
le = LabelEncoder()
y_enc = le.fit_transform(y)
```

(continues on next page)

(continued from previous page)

```
x = df[feature_columns]
x_train, x_test, y_train, y_test = train_test_split(x, y_enc)

# train an EBM model
model = ExplainableBoostingClassifier(
    feature_types=['continuous', 'continuous', 'continuous', 'nominal'],
)
model.fit(x_train, y_train)
```

Then you can convert it to ONNX in a single function call:

```
import onnx
import ebm2onnx

onnx_model = ebm2onnx.to_onnx(
    model,
    ebm2onnx.get_dtype_from_pandas(x_train),
)
onnx.save_model(onnx_model, 'ebm_model.onnx')
```

If your dataset is not a pandas dataframe, you can provide the features' types directly:

```
import ebm2onnx

onnx_model = ebm2onnx.to_onnx(
    model,
    dtype={
        'Age': 'double',
        'Fare': 'double',
        'Pclass': 'int',
        'Embarked': 'str',
    }
)
onnx.save_model(onnx_model, 'ebm_model.onnx')
```

## 1.3 Try it live

- You can live test the [model conversion](#).
- You can live test [local explanations](#).

## 1.4 Supporting organizations

The following organizations are supporting Ebm2onnx:

- [SoftAtHome](#): Main supporter of Ebm2onnx development.
- [InterpretML](#): Ebm2onnx is hosted under the umbrella of the InterpretML organization.





## INSTALLATION

### 2.1 Stable release

To install `ebm2onnx`, run this command in your terminal:

```
$ pip install ebm2onnx
```

This is the preferred method to install `ebm2onnx`, as it will always install the most recent stable release.

If you don't have `pip` installed, this [Python installation guide](#) can guide you through the process.

### 2.2 From sources

The sources for `ebm2onnx` can be downloaded from the [Github repo](#).

You can either clone the public repository:

```
$ git clone git://github.com/interpretml/ebm2onnx.git
```

Or download the [tarball](#):

```
$ curl -OJL https://github.com/interpretml/ebm2onnx/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```



---

## CHAPTER THREE

---

### USAGE

To use ebm2onnx in a project:

```
import ebm2onnx
```



## REFERENCE

Top-level package for ebm2onnx.

`ebm2onnx.get_dtype_from_pandas(df)`

Infers the features names and types from a pandas dataframe

### Example

```
>>>import ebm2onnx >>> >>>dtype = ebm2onnx.get_dtype_from_pandas(my_df)
```

#### Parameters

**df** – A pandas dataframe

#### Returns

A dict that can be used as the type argument of the `to_onnx` function.

`ebm2onnx.to_onnx(model, dtype, name='ebm', predict_proba=False, explain=False, target_opset=None, prediction_name='prediction', probabilities_name='probabilities', explain_name='scores')`

Converts an EBM model to ONNX.

The returned model contains one to three output. The first output is always the prediction, and is named “prediction”. If `predict_proba` is set to True, then another output named “probabilities” is added. If `explain` is set to True, then another output named “scores” is added.

#### Parameters

- **model** – The EBM model, trained with `interpretml`
- **dtype** – A dict containing the type of each input feature. Types are expressed as strings, the following values are supported: float, double, int, str.
- **name** – [Optional] The name of the model
- **predict\_proba** – [Optional] For classification models, output prediction probabilities instead of class
- **explain** – [Optional] Adds an additional output with the score per feature per class
- **target\_opset** – [Optional] The target onnx opset version to use

#### Returns

An ONNX model.



## INDICES AND TABLES

- genindex
- modindex
- search



## PYTHON MODULE INDEX

### e

ebm2onnx, [9](#)



## INDEX

### E

`ebm2onnx`  
    module, [9](#)

### G

`get_dtype_from_pandas()` (*in module `ebm2onnx`*), [9](#)

### M

module  
    `ebm2onnx`, [9](#)

### T

`to_onnx()` (*in module `ebm2onnx`*), [9](#)