
EXperimental PLANning Documentation

Release 0.0.0

properallan

Aug 17, 2023

CONTENTS:

1	Installation	3
2	Documentation	5
2.1	explann	5
2.2	Examples	7
3	Indices and tables	63
	Python Module Index	65
	Index	67

Python package for EXperimental PLANNing course. It implements classes to build and analyse statistical (mainly focused on factorial) models.

Statistical models are built by *statsmodels*.

INSTALLATION

If you have *git* and installed, just clone this repository

```
git clone https://github.com/properallan/explann``
```

move to the root folder

```
cd explann
```

then

```
pip install -e .
```


DOCUMENTATION

The documentation is available at <https://explann.readthedocs.io/en/latest/>

2.1 explann

2.1.1 explann package

Subpackages

Submodules

explann.dataio module

class explann.dataio.**BaseImport**(*data: DataFrame = None, start_index: int = 1*)

Bases: object

Base class for importing data from a file or string.

data : pandas.DataFrame Data to be imported.

data

Type

pandas.DataFrame

parsed_data

Type

pandas.DataFrame

levels

Type

pandas.DataFrame

delimiter

Type

str

parse_levels(*data*)

Parse levels from a pandas.DataFrame.

parse_levels_from_string(data, delimiter)

Parse levels from a string.

parse_levels_from_xlsx(path, sheet_name, index_col)

Parse levels from an xlsx file.

parse_levels(data: DataFrame = None)

parse_levels_from_string(data: str = None, delimiter: str = None)

parse_levels_from_xlsx(data: str = None, sheet_name: str = None, index_col: int = 0)

class explann.dataio.**ImportString**(data: str = None, levels: str = None, delimiter: str = '\s', engine: str = 'python', start_index: int = 1, **kwargs)

Bases: [BaseImport](#)

class explann.dataio.**ImportXLSX**(path: str = None, data_sheet: str = 0, levels_sheet: str = None, start_index: int = 1, **kwargs)

Bases: [BaseImport](#)

explann.models module

class explann.models.**BaseModel**

Bases: object

class explann.models.**FactorialModel**(data: ~pandas.core.frame.DataFrame = None, functions: str | list | tuple = None, statsmodel: object = <bound method Model.from_formula of <class 'statsmodels.regression.linear_model.OLS'>>, levels: ~pandas.core.frame.DataFrame = None, **fit_kwargs)

Bases: [BaseModel](#)

anova(function: str | list | tuple = None)

build_significant_models(function: str | list | tuple = None, alpha: float = 0.05, use_anova: bool = False)

decode_variables(variables, levels: DataFrame = None)

property dependent_variables

encode_variables(variables, levels: DataFrame = None)

fit(data: ~pandas.core.frame.DataFrame = None, functions: str | list | tuple = None, statsmodel: object = <bound method Model.from_formula of <class 'statsmodels.regression.linear_model.OLS'>>, **fit_kwargs)

property function_names

get_categorical_functions()

get_categorical_model()

get_significant_model_functions(function: str | list | tuple = None, alpha: float = 0.05, use_anova: bool = False)

```

get_significant_terms(function: str | list | tuple = None, alpha: float = 0.05)

property independent_variables

lack_of_fit(function: str | list | tuple = None, baseline: BaseModel = None, alpha: float = 0.05)

predict(function, variables)

predict_rescaled(function, variables, levels: DataFrame = None)

print_equation(function: str | list | tuple = None, alpha: float = 0.05, precision: int = 4)

summary(function: str | list | tuple = None)

working_lack_of_fit(function: str | list | tuple = None, baseline: BaseModel = None)

```

```
explann.models.add_categorical(formula)
```

Module contents

2.2 Examples

2.2.1 Begginer's Tutorial

Reading Data from String

Data can be read from a string, by using the `ImportString` class of `explain.dataio`. The default delimiter is `\s` the special caractere for whitespaces.

```
[1]: from explann.dataio import ImportString
```

```

data_string = """
Observação  Dureza  Temperatura
1   137      220
2   137      220
3   137      220
4   136      220
5   135      220
6   135      225
7   133      225
8   132      225
9   133      225
10  133      225
11  128      230
12  124      230
13  126      230
14  129      230
15  126      230
16  122      235
17  122      235
18  122      235
19  119      235
20  122      235

```

(continues on next page)

(continued from previous page)

"""

```
data_reader_string = ImportString(data=data_string, delimiter="\s")
```

data_reader object stores the provided data in its .data attribute

```
[2]: data_reader_string.data
```

```
[2]:
```

	Observação	Dureza	Temperatura
0	1	137	220
1	2	137	220
2	3	137	220
3	4	136	220
4	5	135	220
5	6	135	225
6	7	133	225
7	8	132	225
8	9	133	225
9	10	133	225
10	11	128	230
11	12	124	230
12	13	126	230
13	14	129	230
14	15	126	230
15	16	122	235
16	17	122	235
17	18	122	235
18	19	119	235
19	20	122	235

Read Data from Excel file

Data can also be read from a .xlsx file (Excel extension). To do so, use the ImportXLSX class of explain.dataio. Default is to read the first Sheet, otherwise desired provide the additional argument sheet_name

```
[3]: from explann.dataio import ImportXLSX
```

```
data_reader_xlsx = ImportXLSX(path="../../../data/paper_data_24.xlsx")
```

```
data_reader_xlsx.data
```

```
[3]:
```

	U	A	P	Y	F	C	B
0	-1	-1	-1	-1	39	1.328	170
1	1	-1	-1	-1	87	1.699	122
2	-1	1	-1	-1	48	1.332	473
3	1	1	-1	-1	71	1.979	511
4	-1	-1	1	-1	43	1.458	156
5	1	-1	1	-1	84	2.189	204
6	-1	1	1	-1	45	1.343	385
7	1	1	1	-1	112	1.707	288
8	-1	-1	-1	1	19	1.257	114
9	1	-1	-1	1	146	2.148	116

(continues on next page)

(continued from previous page)

10	-1	1	-1	1	50	1.592	244
11	1	1	-1	1	92	1.726	126
12	-1	-1	1	1	107	1.203	72
13	1	-1	1	1	172	2.261	210
14	-1	1	1	1	62	1.434	234
15	1	1	1	1	82	1.848	154
16	0	0	0	0	75	1.726	223
17	0	0	0	0	70	1.782	219
18	0	0	0	0	89	1.753	226

Dealing With Leveled Data

Any importer has the functionality to convert levels of an factorial to respective values. For this task a table containing the values associated with each level should be passed, either as a `string` or a `.xlsx` file in the same way as the main data table.

```
[4]: levels_string = """
Levels;U;A;P;Y
-1;0.15;0.7; 0.40;0.13
0; 0.30;1.4; 0.75;0.26
1; 0.45;2.1; 1.10;0.38
"""

levels_reader = ImportString(
    data = levels_string,
    delimiter = ";",
    index_col = 0 # should pass the column name or index containing the level.
)
levels_reader.data
```

```
[4]:      U      A      P      Y
Levels
-1      0.15  0.7   0.40  0.13
0       0.30  1.4   0.75  0.26
1       0.45  2.1   1.10  0.38
```

The same data should be imported from a `.xlsx` file.

```
[5]: levels_reader_xlsx = ImportXLSX(
    path="../../data/paper_data_24.xlsx",
    sheet_name="Levels",
    index_col=0,
)
levels_reader_xlsx.data
```

```
[5]:      U      A      P      Y
Levels
-1      0.15  0.7   0.40  0.13
0       0.30  1.4   0.75  0.26
1       0.45  2.1   1.10  0.38
```

The data reader `parse_levels` accept a `pd.DataFrame` constructed from any one of the methods above, you can pass string or `.xlsx` files to the methods `parse_levels_from_string` and `parse_levels_from_xlsx`.

```
[6]: # passing a pd.DataFrame
data_reader_xlsx.parse_levels(
    data = levels_reader.data
)
```

```
[7]: data_reader_xlsx.data
```

```
[7]:
```

	U	A	P	Y	F	C	B
0	0.15	0.7	0.40	0.13	39	1.328	170
1	0.45	0.7	0.40	0.13	87	1.699	122
2	0.15	2.1	0.40	0.13	48	1.332	473
3	0.45	2.1	0.40	0.13	71	1.979	511
4	0.15	0.7	1.10	0.13	43	1.458	156
5	0.45	0.7	1.10	0.13	84	2.189	204
6	0.15	2.1	1.10	0.13	45	1.343	385
7	0.45	2.1	1.10	0.13	112	1.707	288
8	0.15	0.7	0.40	0.38	19	1.257	114
9	0.45	0.7	0.40	0.38	146	2.148	116
10	0.15	2.1	0.40	0.38	50	1.592	244
11	0.45	2.1	0.40	0.38	92	1.726	126
12	0.15	0.7	1.10	0.38	107	1.203	72
13	0.45	0.7	1.10	0.38	172	2.261	210
14	0.15	2.1	1.10	0.38	62	1.434	234
15	0.45	2.1	1.10	0.38	82	1.848	154
16	0.30	1.4	0.75	0.26	75	1.726	223
17	0.30	1.4	0.75	0.26	70	1.782	219
18	0.30	1.4	0.75	0.26	89	1.753	226

```
[8]: # passing a string
data_reader_xlsx.parse_levels_from_string(
    data = levels_string,
    delimiter=";"
)

# passing a path
data_reader_xlsx.parse_levels_from_xlsx(
    data = "../data/paper_data_24.xlsx",
    sheet_name = "Levels",
    index_col=0,
)
```

```
[9]: data_reader_xlsx.data
```

```
[9]:
```

	U	A	P	Y	F	C	B
0	0.15	0.7	0.40	0.13	39	1.328	170
1	0.45	0.7	0.40	0.13	87	1.699	122
2	0.15	2.1	0.40	0.13	48	1.332	473
3	0.45	2.1	0.40	0.13	71	1.979	511
4	0.15	0.7	1.10	0.13	43	1.458	156
5	0.45	0.7	1.10	0.13	84	2.189	204
6	0.15	2.1	1.10	0.13	45	1.343	385
7	0.45	2.1	1.10	0.13	112	1.707	288

(continues on next page)

(continued from previous page)

8	0.15	0.7	0.40	0.38	19	1.257	114
9	0.45	0.7	0.40	0.38	146	2.148	116
10	0.15	2.1	0.40	0.38	50	1.592	244
11	0.45	2.1	0.40	0.38	92	1.726	126
12	0.15	0.7	1.10	0.38	107	1.203	72
13	0.45	0.7	1.10	0.38	172	2.261	210
14	0.15	2.1	1.10	0.38	62	1.434	234
15	0.45	2.1	1.10	0.38	82	1.848	154
16	0.30	1.4	0.75	0.26	75	1.726	223
17	0.30	1.4	0.75	0.26	70	1.782	219
18	0.30	1.4	0.75	0.26	89	1.753	226

The results are the same, data attribute has its values parsed to the corresponding index levels for each variable as described in the `levels_reades_<type>.data` table.

```
[10]: data_reader_xlsx.data
```

```
[10]:
```

	U	A	P	Y	F	C	B
0	0.15	0.7	0.40	0.13	39	1.328	170
1	0.45	0.7	0.40	0.13	87	1.699	122
2	0.15	2.1	0.40	0.13	48	1.332	473
3	0.45	2.1	0.40	0.13	71	1.979	511
4	0.15	0.7	1.10	0.13	43	1.458	156
5	0.45	0.7	1.10	0.13	84	2.189	204
6	0.15	2.1	1.10	0.13	45	1.343	385
7	0.45	2.1	1.10	0.13	112	1.707	288
8	0.15	0.7	0.40	0.38	19	1.257	114
9	0.45	0.7	0.40	0.38	146	2.148	116
10	0.15	2.1	0.40	0.38	50	1.592	244
11	0.45	2.1	0.40	0.38	92	1.726	126
12	0.15	0.7	1.10	0.38	107	1.203	72
13	0.45	0.7	1.10	0.38	172	2.261	210
14	0.15	2.1	1.10	0.38	62	1.434	234
15	0.45	2.1	1.10	0.38	82	1.848	154
16	0.30	1.4	0.75	0.26	75	1.726	223
17	0.30	1.4	0.75	0.26	70	1.782	219
18	0.30	1.4	0.75	0.26	89	1.753	226

Original data is retained in a `raw_data` attribute

```
[11]: data_reader_xlsx.raw_data
```

```
[11]:
```

	U	A	P	Y	F	C	B
0	-1	-1	-1	-1	39	1.328	170
1	1	-1	-1	-1	87	1.699	122
2	-1	1	-1	-1	48	1.332	473
3	1	1	-1	-1	71	1.979	511
4	-1	-1	1	-1	43	1.458	156
5	1	-1	1	-1	84	2.189	204
6	-1	1	1	-1	45	1.343	385
7	1	1	1	-1	112	1.707	288
8	-1	-1	-1	1	19	1.257	114
9	1	-1	-1	1	146	2.148	116

(continues on next page)

(continued from previous page)

10	-1	1	-1	1	50	1.592	244
11	1	1	-1	1	92	1.726	126
12	-1	-1	1	1	107	1.203	72
13	1	-1	1	1	172	2.261	210
14	-1	1	1	1	62	1.434	234
15	1	1	1	1	82	1.848	154
16	0	0	0	0	75	1.726	223
17	0	0	0	0	70	1.782	219
18	0	0	0	0	89	1.753	226

Build a Factorial Model

To build a factorial model for the data `explann` implemented the class `FactorialModel`. The arguments are the data and the functions. Functions are passed as python dictionaries, the keys are the function names, in this example "F", "CM" and "B". This names should be any string, this allow to create any number of models by addressing different names.

The dictionary values are also string containing the model equations, the syntax follow `patsy` standards. The left hand side (lhs) contains the dependent variables and the right hand side (rhs) the independent terms. The lhs and rhs are separated by "~" caractere.

```
[13]: from explann.models import FactorialModel
```

```
fm = FactorialModel(
    data=data_reader_xlsx.data,
    functions=
    {
        "F" : "F ~ U * A * P * Y",
        "CM" : "CM ~ U * A * P * Y",
        "B" : "B ~ U * A * P * Y"
    }
)
```

```
-----
NameError                                Traceback (most recent call last)
File ~/micromamba/envs/explann/lib/python3.11/site-packages/patsy/compat.py:36, in call_
↳ and_wrap_exc(msg, origin, f, *args, **kwargs)
    35 try:
--> 36     return f(*args, **kwargs)
    37 except Exception as e:

File ~/micromamba/envs/explann/lib/python3.11/site-packages/patsy/eval.py:169, in
↳ EvalEnvironment.eval(self, expr, source_name, inner_namespace)
    168 code = compile(expr, source_name, "eval", self.flags, False)
--> 169 return eval(code, {}, VarLookupDict([inner_namespace]
    170                                     + self._namespaces))
```

```
File <string>:1
```

```
NameError: name 'CM' is not defined
```

The above exception was the direct cause of the following exception:

(continues on next page)

(continued from previous page)

```

PatsyError                                Traceback (most recent call last)
Cell In[13], line 3
      1 from explann.models import FactorialModel
----> 3 fm = FactorialModel(
      4     data=data_reader.xlsx.data,
      5     functions=
      6     {
      7         "F" : "F ~ U * A * P * Y",
      8         "CM" : "CM ~ U * A * P * Y",
      9         "B" : "B ~ U * A * P * Y"
     10     }
     11 )

File ~/Dropbox/local/github/explann/explann/models.py:34, in FactorialModel.__init__
    ↪(self, data, functions, statsmodel)
     31 self.statsmodel = statsmodel
     32 self.functions = functions
--> 34 self.model = self.fit(
     35     data=data,
     36     functions=functions,
     37     statsmodel=statsmodel
     38 )

File ~/Dropbox/local/github/explann/explann/models.py:56, in FactorialModel.fit(self,
    ↪data, functions, statsmodel)
     54     function_name = splited[0].strip()
     55     function_body = splited[1].strip()
--> 56     model_dict[key] = self.statsmodel(function, data).fit()
     57 else:
     58     print('must provide function formulas')

File ~/micromamba/envs/explann/lib/python3.11/site-packages/statsmodels/base/model.py:
    ↪203, in Model.from_formula(cls, formula, data, subset, drop_cols, *args, **kwargs)
     200 if missing == 'none': # with patsy it's drop or raise. let's raise.
     201     missing = 'raise'
--> 203 tmp = handle_formula_data(data, None, formula, depth=eval_env,
     204                             missing=missing)
     205 ((endog, exog), missing_idx, design_info) = tmp
     206 max_endog = cls._formula_max_endog

File ~/micromamba/envs/explann/lib/python3.11/site-packages/statsmodels/formula/
    ↪formulatools.py:63, in handle_formula_data(Y, X, formula, depth, missing)
     61 else:
     62     if data_util._is_using_pandas(Y, None):
--> 63         result = dmatrixes(formula, Y, depth, return_type='dataframe',
     64                             NA_action=na_action)
     65     else:
     66         result = dmatrixes(formula, Y, depth, return_type='dataframe',
     67                             NA_action=na_action)

File ~/micromamba/envs/explann/lib/python3.11/site-packages/patsy/highlevel.py:309, in

```

(continues on next page)

(continued from previous page)

```

->dmatrices(formula_like, data, eval_env, NA_action, return_type)
    299 """Construct two design matrices given a formula_like and data.
    300
    301 This function is identical to :func:`dmatrix`, except that it requires
    (...)
    306 See :func:`dmatrix` for details.
    307 """
    308 eval_env = EvalEnvironment.capture(eval_env, reference=1)
--> 309 (lhs, rhs) = _do_highlevel_design(formula_like, data, eval_env,
    310                                     NA_action, return_type)
    311 if lhs.shape[1] == 0:
    312     raise PatsyError("model is missing required outcome variables")

File ~/micromamba/envs/explann/lib/python3.11/site-packages/patsy/highlevel.py:164, in _
->do_highlevel_design(formula_like, data, eval_env, NA_action, return_type)
    162 def data_iter_maker():
    163     return iter([data])
--> 164 design_infos = _try_incr_builders(formula_like, data_iter_maker, eval_env,
    165                                     NA_action)
    166 if design_infos is not None:
    167     return build_design_matrices(design_infos, data,
    168                                   NA_action=NA_action,
    169                                   return_type=return_type)

File ~/micromamba/envs/explann/lib/python3.11/site-packages/patsy/highlevel.py:66, in _
->try_incr_builders(formula_like, data_iter_maker, eval_env, NA_action)
    64 if isinstance(formula_like, ModelDesc):
    65     assert isinstance(eval_env, EvalEnvironment)
--> 66     return design_matrix_builders([formula_like.lhs_termlist,
    67                                     formula_like.rhs_termlist],
    68                                     data_iter_maker,
    69                                     eval_env,
    70                                     NA_action)
    71 else:
    72     return None

File ~/micromamba/envs/explann/lib/python3.11/site-packages/patsy/build.py:693, in _
->design_matrix_builders(termlists, data_iter_maker, eval_env, NA_action)
    689 factor_states = _factors_memorize(all_factors, data_iter_maker, eval_env)
    690 # Now all the factors have working eval methods, so we can evaluate them
    691 # on some data to find out what type of data they return.
    692 (num_column_counts,
--> 693  cat_levels_contrasts) = _examine_factor_types(all_factors,
    694                                                  factor_states,
    695                                                  data_iter_maker,
    696                                                  NA_action)
    697 # Now we need the factor infos, which encapsulate the knowledge of
    698 # how to turn any given factor into a chunk of data:
    699 factor_infos = {}

File ~/micromamba/envs/explann/lib/python3.11/site-packages/patsy/build.py:443, in _
->examine_factor_types(factors, factor_states, data_iter_maker, NA_action)

```

(continues on next page)

(continued from previous page)

```

441 for data in data_iter_maker():
442     for factor in list(examine_needed):
--> 443         value = factor.eval(factor_states[factor], data)
444         if factor in cat_sniffers or guess_categorical(value):
445             if factor not in cat_sniffers:

File ~/micromamba/envs/explann/lib/python3.11/site-packages/patsy/eval.py:568, in_
EvalFactor.eval(self, memorize_state, data)
567 def eval(self, memorize_state, data):
--> 568     return self._eval(memorize_state["eval_code"],
569                         memorize_state,
570                         data)

File ~/micromamba/envs/explann/lib/python3.11/site-packages/patsy/eval.py:551, in_
EvalFactor._eval(self, code, memorize_state, data)
549 def _eval(self, code, memorize_state, data):
550     inner_namespace = VarLookupDict([data, memorize_state["transforms"]])
--> 551     return call_and_wrap_exc("Error evaluating factor",
552                             self,
553                             memorize_state["eval_env"].eval,
554                             code,
555                             inner_namespace=inner_namespace)

File ~/micromamba/envs/explann/lib/python3.11/site-packages/patsy/compat.py:43, in call_
and_wrap_exc(msg, origin, f, *args, **kwargs)
39     new_exc = PatsyError("%s: %s: %s"
40                          % (msg, e.__class__.__name__, e),
41                          origin)
42     # Use 'exec' to hide this syntax from the Python 2 parser:
--> 43     exec("raise new_exc from e")
44 else:
45     # In python 2, we just let the original exception escape -- better
46     # than destroying the traceback. But if it's a PatsyError, we can
47     # at least set the origin properly.
48     if isinstance(e, PatsyError):

File <string>:1

PatsyError: Error evaluating factor: NameError: name 'CM' is not defined
CM ~ U * A * P * Y
^^

```

[]:

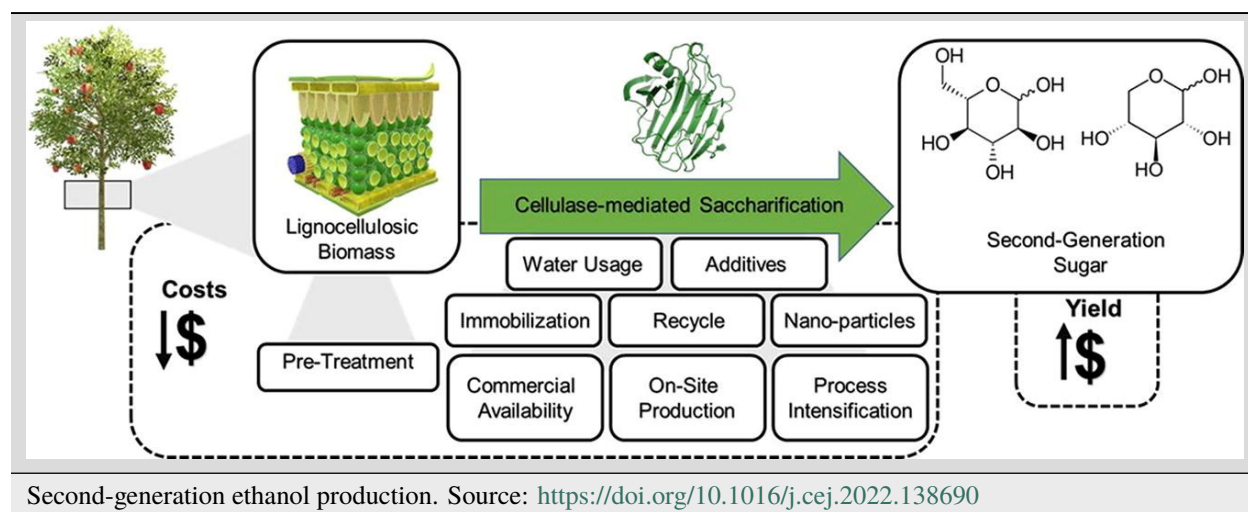
2.2.2 Case Study

Nitrogen Source Optimization for Cellulase Production by *Penicillium funiculosum*, using a Sequential Experimental Design Methodology and the Desirability Function

The chosen [article](#) for the case study suggests employing a sequential experimental design methodology to enhance cellulase production efficiency. Lignocellulosic biomass, being one of the most abundant materials globally, holds immense potential for biomass production. However, the widespread application of lignocellulosic biomass for biofuel production faces constraints due to the considerable expenses associated with acquiring the necessary enzymes for the chemical process of deriving second-generation sugars. Consequently, the optimization of enzyme production emerges as a pivotal focus within the energy industry's interests.

The Chosen Article

Maeda, R.N., da Silva, M.M.P., Santa Anna, L.M.M. et al. Nitrogen Source Optimization for Cellulase Production by *Penicillium funiculosum*, using a Sequential Experimental Design Methodology and the Desirability Function. *Appl Biochem Biotechnol* 161, 411–422 (2010)



Experimental Methods

Cellulase production by *Penicillium funiculosum* was assessed through the measurement of cellulase activity, quantified in terms of activity units (U), indicating the enzymatic extract required to liberate 1 mol of sugars per minute.

Three distinct methods were employed to quantify cellulase activity, serving as the variables under consideration for optimization:

- Filter Paper Assay (FPase)
- Carboxymethylcellulose Assay (CMCase)
- Cellobiose Assay (-Glucosidase)

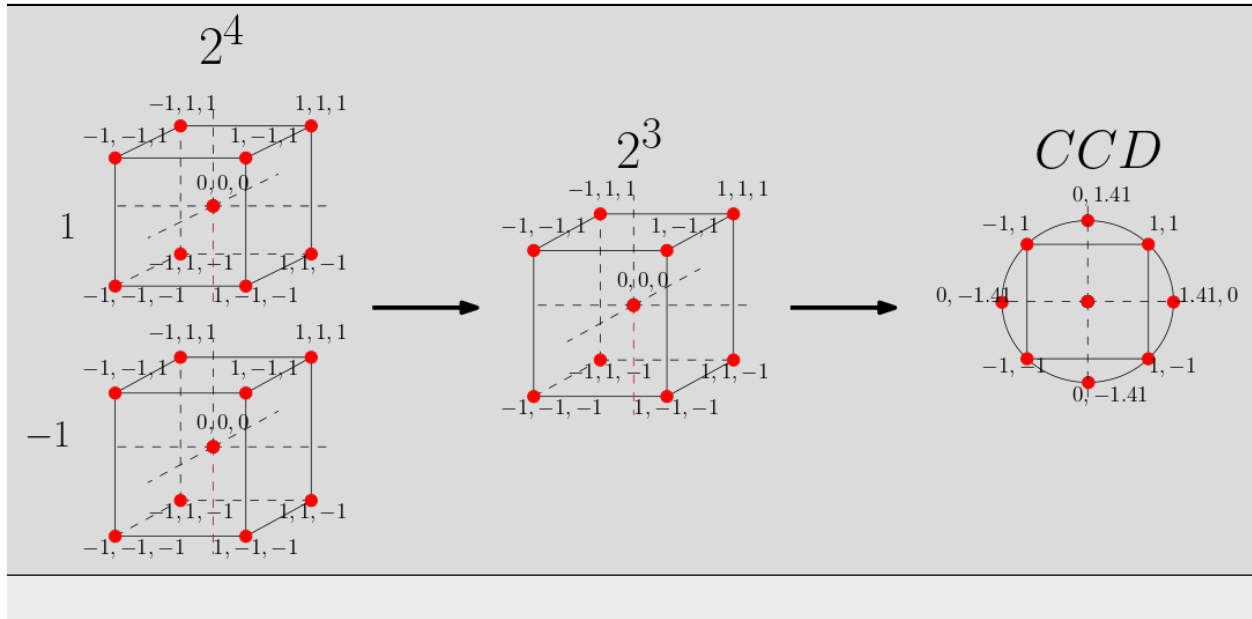
The dependent variables in this study encompass the concentrations of various nitrogen sources present within the samples, namely:

- Urea
- Ammonium Sulfate
- Peptone

- Yeast Extract

The experimental samples consist of conical flasks, each containing 200 mL of a production medium. This medium features varying concentrations of urea, ammonium sulfate, peptone, and yeast extract, all acting as nitrogen sources. It's worth noting that the carbon source remains constant throughout the experiments, maintaining a concentration of 15 g/L of partially delignified Cellulignin. Additionally, the spore suspension of *Penicillium funiculosum* ATCC 11797 maintains a consistent value of 10^6 conidia/mL.

Sequential Experimenta Design



The sequential experimental design involves a series of meticulously crafted iterations, wherein each step entails the elimination of an insignificant variable.

This systematic approach begins with a 2^4 factorial design, from which the least influential variable is identified and excluded based on Pareto plot analysis of the effects observed. The ensuing stage employs the remaining three variables within a 2^3 experimental design, following a similar process of identifying and omitting the least impactful variable. Ultimately, the final stage entails the execution of a central composite design, employing the last two variables that have emerged from the iterative refinement process.

2.2.3 Factorial 2^4

Factor	Min -1	CP 0	Max +1
Urea (g/L)	0.15	0.30	0.45
Ammonium sulfate (g/L)	0.70	1.40	2.10
Peptone (g/L)	0.40	0.75	1.10
Yeast extract (g/L)	0.13	0.26	0.38
<i>Min</i> minimum value, <i>CP</i> center point, <i>Max</i> maximum value.			
Levels for independent variables for the 2^4 experimental design.			

Run	Nitrogen sources				Activities (U/L)		
	Urea	Ammonium sulfate	Peptone	Yeast extract	FPase $R^2=0.836$	CMCase $R^2=0.873$	β -glucosidase $R^2=0.910$
1	-1	-1	-1	-1	39	1,328	170
2	+1	-1	-1	-1	87	1,699	122
3	-1	+1	-1	-1	48	1,332	473
4	+1	+1	-1	-1	71	1,979	511
5	-1	-1	+1	-1	43	1,458	156
6	+1	-1	+1	-1	84	2,189	204
7	-1	+1	+1	-1	45	1,343	385
8	+1	+1	+1	-1	112	1,707	288
9	-1	-1	-1	+1	19	1,257	114
10	+1	-1	-1	+1	146	2,148	116
11	-1	+1	-1	+1	50	1,592	244
12	+1	+1	-1	+1	92	1,726	126
13	-1	-1	+1	+1	107	1,203	72
14	+1	-1	+1	+1	172	2,261	210
15	-1	+1	+1	+1	62	1,434	234
16	+1	+1	+1	+1	82	1,848	154
17 (C)	0	0	0	0	75	1,726	223
18 (C)	0	0	0	0	70	1,782	219
19 (C)	0	0	0	0	89	1,753	226

Experimental results of each run of the 2^4 design.

2.2.4 explann hands-on on 2^4 design

A python package (`explann` <<https://github.com/properallan/explann/>> `__`) was developed to assist design and statistical analysis of experiments.

All the following source code is hosted on github <https://github.com/properallan/explann/>. Both the table of levels and the experimental results can be easily imported to the `explann` package. There are functions to import data in string format or even `xlsx` file format. The `explann` package can assist also in the creation of the experimental design.

`TwoLevelFactorial` implements a generic 2^n factorial design, with n variables defined as a dictionary containing the variable name and range.

```
[1]: from explann.doe import TwoLevelFactorial

f2b4 = TwoLevelFactorial(
    variables = {
        'U': (0.15, 0.45),
        'A': (0.70, 2.10),
        'P': (0.40, 1.10),
        'Y': (0.13, 0.38)
    },
    central_points=3
)
```

Once instantiated this class build the `doe` table, storage as an object attribute as a `pandas DataFrame`.

```
[2]: f2b4.doe
```

```
[2]:
```

	U	A	P	Y
Index				
1	-1.0	-1.0	-1.0	-1.0
2	1.0	-1.0	-1.0	-1.0
3	-1.0	1.0	-1.0	-1.0
4	1.0	1.0	-1.0	-1.0
5	-1.0	-1.0	1.0	-1.0
6	1.0	-1.0	1.0	-1.0
7	-1.0	1.0	1.0	-1.0
8	1.0	1.0	1.0	-1.0
9	-1.0	-1.0	-1.0	1.0
10	1.0	-1.0	-1.0	1.0
11	-1.0	1.0	-1.0	1.0
12	1.0	1.0	-1.0	1.0
13	-1.0	-1.0	1.0	1.0
14	1.0	-1.0	1.0	1.0
15	-1.0	1.0	1.0	1.0
16	1.0	1.0	1.0	1.0
17	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0

A table of the levels are also automatically created.

```
[3]: f2b4.levels
```

```
[3]:
```

	U	A	P	Y
Levels				

(continues on next page)

(continued from previous page)

-1.0	0.15	0.7	0.40	0.130
0.0	0.30	1.4	0.75	0.255
1.0	0.45	2.1	1.10	0.380

The doe table can be complemented with the response variables. For this task either `ImportString` or `ImportXLSX` can be used to load data.

```
[4]: from explann.dataio import ImportString
```

```
f2b4_results = ImportString(data="""
F,CM,B
39,1.328,170
87,1.699,122
48,1.332,473
71,1.979,511
43,1.458,156
84,2.189,204
45,1.343,385
112,1.707,288
19,1.257,114
146,2.148,116
50,1.592,244
92,1.726,126
107,1.203,72
172,2.261,210
62,1.434,234
82,1.848,154
75,1.726,223
70,1.782,219
89,1.753,226
""",
delimiter=',')
```

```
f2b4_results.data
```

```
[4]:      F      CM      B
1      39    1.328    170
2      87    1.699    122
3      48    1.332    473
4      71    1.979    511
5      43    1.458    156
6      84    2.189    204
7      45    1.343    385
8     112    1.707    288
9      19    1.257    114
10    146    2.148    116
11     50    1.592    244
12     92    1.726    126
13    107    1.203     72
14    172    2.261    210
15     62    1.434    234
16     82    1.848    154
17     75    1.726    223
```

(continues on next page)

(continued from previous page)

```
18  70  1.782  219
19  89  1.753  226
```

The data is merged with the doe table using `append_results` method.

```
[5]: f2b4.append_results(results=f2b4_results.data)
     f2b4.doe
```

```
[5]:
```

	U	A	P	Y	F	CM	B
Index							
1	-1.0	-1.0	-1.0	-1.0	39	1.328	170
2	1.0	-1.0	-1.0	-1.0	87	1.699	122
3	-1.0	1.0	-1.0	-1.0	48	1.332	473
4	1.0	1.0	-1.0	-1.0	71	1.979	511
5	-1.0	-1.0	1.0	-1.0	43	1.458	156
6	1.0	-1.0	1.0	-1.0	84	2.189	204
7	-1.0	1.0	1.0	-1.0	45	1.343	385
8	1.0	1.0	1.0	-1.0	112	1.707	288
9	-1.0	-1.0	-1.0	1.0	19	1.257	114
10	1.0	-1.0	-1.0	1.0	146	2.148	116
11	-1.0	1.0	-1.0	1.0	50	1.592	244
12	1.0	1.0	-1.0	1.0	92	1.726	126
13	-1.0	-1.0	1.0	1.0	107	1.203	72
14	1.0	-1.0	1.0	1.0	172	2.261	210
15	-1.0	1.0	1.0	1.0	62	1.434	234
16	1.0	1.0	1.0	1.0	82	1.848	154
17	0.0	0.0	0.0	0.0	75	1.726	223
18	0.0	0.0	0.0	0.0	70	1.782	219
19	0.0	0.0	0.0	0.0	89	1.753	226

This experimental planning can be easily saved in `.xlsx` format. `doe` raw data is stored in the first sheet and the levels data is stored in a separate sheet named `levels`.

```
[6]: f2b4.save_excel('../data/f2b4.xlsx')
```

The `ImportXLSX` assist the import of data from excel file, this could be filled by hand or generated in `explann` as described above.

```
[7]: from explann.dataio import ImportXLSX
```

```
f2b4_from_excel = ImportXLSX(
    path = '../data/f2b4.xlsx',
    levels_sheet = 'levels'
)
```

```
f2b4_from_excel.data
```

```
[7]:
```

	Index	U	A	P	Y	F	CM	B
1	1	-1	-1	-1	-1	39	1.328	170
2	2	1	-1	-1	-1	87	1.699	122
3	3	-1	1	-1	-1	48	1.332	473
4	4	1	1	-1	-1	71	1.979	511
5	5	-1	-1	1	-1	43	1.458	156
6	6	1	-1	1	-1	84	2.189	204

(continues on next page)

(continued from previous page)

7	7	-1	1	1	-1	45	1.343	385
8	8	1	1	1	-1	112	1.707	288
9	9	-1	-1	-1	1	19	1.257	114
10	10	1	-1	-1	1	146	2.148	116
11	11	-1	1	-1	1	50	1.592	244
12	12	1	1	-1	1	92	1.726	126
13	13	-1	-1	1	1	107	1.203	72
14	14	1	-1	1	1	172	2.261	210
15	15	-1	1	1	1	62	1.434	234
16	16	1	1	1	1	82	1.848	154
17	17	0	0	0	0	75	1.726	223
18	18	0	0	0	0	70	1.782	219
19	19	0	0	0	0	89	1.753	226

If the `levels_sheet` argument is passed, the loaded object will contain an additional table with the levels parsed.

```
[8]: f2b4_from_excel.parsed_data
```

```
[8]:
```

	Index	U	A	P	Y	F	CM	B
1	1	0.45	0.7	0.40	0.130	39	1.328	170
2	2	0.15	2.1	0.40	0.130	87	1.699	122
3	3	0.45	2.1	0.40	0.130	48	1.332	473
4	4	0.15	0.7	1.10	0.130	71	1.979	511
5	5	0.45	0.7	1.10	0.130	43	1.458	156
6	6	0.15	2.1	1.10	0.130	84	2.189	204
7	7	0.45	2.1	1.10	0.130	45	1.343	385
8	8	0.15	0.7	0.40	0.380	112	1.707	288
9	9	0.45	0.7	0.40	0.380	19	1.257	114
10	10	0.15	2.1	0.40	0.380	146	2.148	116
11	11	0.45	2.1	0.40	0.380	50	1.592	244
12	12	0.15	0.7	1.10	0.380	92	1.726	126
13	13	0.45	0.7	1.10	0.380	107	1.203	72
14	14	0.15	2.1	1.10	0.380	172	2.261	210
15	15	0.45	2.1	1.10	0.380	62	1.434	234
16	16	0.30	1.4	0.75	0.255	82	1.848	154
17	17	0.30	1.4	0.75	0.255	75	1.726	223
18	18	0.30	1.4	0.75	0.255	70	1.782	219
19	19	0.00	0.0	0.00	0.000	89	1.753	226

We can now use the data to build our factorial model. The `functions` attribute is a dictionary, the keys are function names, and the values are function equations, the syntax follows ``patsy`` <https://patsy.readthedocs.io/en/latest/> standardization. Here `U * A * P * Y` stands for all interaction terms between the 4 variables

```
[9]: from explann.models import FactorialModel
```

```
fm2b4 = FactorialModel(
    data = f2b4_from_excel.data,
    functions = {
        "Fso" : "F ~ U * A * P * Y + I(U**2) + I(A**2) + I(P**2) + I(Y**2)",
        "CMso" : "CM ~ U * A * P * Y + I(U**2) + I(A**2) + I(P**2) + I(Y**2)",
        "Bso" : "B ~ U * A * P * Y + I(U**2) + I(A**2) + I(P**2) + I(Y**2)",
        "F" : "F ~ U * A * P * Y",
        "CM" : "CM ~ U * A * P * Y",
```

(continues on next page)

(continued from previous page)

```
"B" : "B ~ U * A * P * Y"}
)
```

The object fm2b4 stores all the fitted models(Ordinary Least Squares) listed in functions, any given model can be retrieved by indexing by function name.

```
[10]: fm2b4['F']
```

```
[10]: <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7fcd1570edf0>
```

An overview of the results can be viewed using the summary method. The method accepts also a given function name as argument, if None is passed a dictionary of summary() is returned.

```
[11]: fm2b4.summary()
```

```
/home/ppiper/micromamba/envs/explann/lib/python3.9/site-packages/scipy/stats/_stats_py.
↳py:1806: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=19
  warnings.warn("kurtosistest only valid for n>=20 ... continuing ")
/home/ppiper/micromamba/envs/explann/lib/python3.9/site-packages/scipy/stats/_stats_py.
↳py:1806: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=19
  warnings.warn("kurtosistest only valid for n>=20 ... continuing ")
/home/ppiper/micromamba/envs/explann/lib/python3.9/site-packages/scipy/stats/_stats_py.
↳py:1806: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=19
  warnings.warn("kurtosistest only valid for n>=20 ... continuing ")
/home/ppiper/micromamba/envs/explann/lib/python3.9/site-packages/scipy/stats/_stats_py.
↳py:1806: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=19
  warnings.warn("kurtosistest only valid for n>=20 ... continuing ")
/home/ppiper/micromamba/envs/explann/lib/python3.9/site-packages/scipy/stats/_stats_py.
↳py:1806: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=19
  warnings.warn("kurtosistest only valid for n>=20 ... continuing ")
/home/ppiper/micromamba/envs/explann/lib/python3.9/site-packages/scipy/stats/_stats_py.
↳py:1806: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=19
  warnings.warn("kurtosistest only valid for n>=20 ... continuing ")
```

```
[11]: {'Fso': <class 'statsmodels.iolib.summary.Summary'>
      ""
```

```

                                OLS Regression Results
=====
Dep. Variable:                  F      R-squared:                  0.992
Model:                        OLS    Adj. R-squared:             0.931
Method:                    Least Squares    F-statistic:                16.16
Date:                Wed, 16 Aug 2023    Prob (F-statistic):          0.0598
Time:                  19:16:43    Log-Likelihood:             -49.032
No. Observations:                19    AIC:                        132.1
Df Residuals:                    2    BIC:                        148.1
Df Model:                      16
Covariance Type:                nonrobust
=====
               coef      std err          t      P>|t|      [0.025      0.975]
-----
Intercept      78.0000      5.686     13.717      0.005     53.534     102.466
U               27.0625      2.462     10.991      0.008     16.468     37.657
A              -8.4375      2.462     -3.427      0.076    -19.032      2.157
U:A            -8.0625      2.462     -3.274      0.082    -18.657      2.532
=====
```

(continues on next page)

(continued from previous page)

P	9.6875	2.462	3.934	0.059	-0.907	20.282
U:P	-2.9375	2.462	-1.193	0.355	-13.532	7.657
A:P	-4.6875	2.462	-1.904	0.197	-15.282	5.907
U:A:P	5.6875	2.462	2.310	0.147	-4.907	16.282
Y	12.5625	2.462	5.102	0.036	1.968	23.157
U:Y	4.6875	2.462	1.904	0.197	-5.907	15.282
A:Y	-11.3125	2.462	-4.594	0.044	-21.907	-0.718
U:A:Y	-8.1875	2.462	-3.325	0.080	-18.782	2.407
P:Y	4.8125	2.462	1.955	0.190	-5.782	15.407
U:P:Y	-7.5625	2.462	-3.071	0.092	-18.157	3.032
A:P:Y	-9.3125	2.462	-3.782	0.063	-19.907	1.282
U:A:P:Y	-0.6875	2.462	-0.279	0.806	-11.282	9.907
I(U ** 2)	0.1719	1.549	0.111	0.922	-6.493	6.837
I(A ** 2)	0.1719	1.549	0.111	0.922	-6.493	6.837
I(P ** 2)	0.1719	1.549	0.111	0.922	-6.493	6.837
I(Y ** 2)	0.1719	1.549	0.111	0.922	-6.493	6.837

Omnibus:	19.262	Durbin-Watson:	2.036
Prob(Omnibus):	0.000	Jarque-Bera (JB):	38.617
Skew:	1.278	Prob(JB):	4.12e-09
Kurtosis:	9.500	Cond. No.	1.96e+30

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The input rank is higher than the number of observations.

[3] The smallest eigenvalue is 2.1e-59. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

```

"""
'CMs0': <class 'statsmodels.iolib.summary.Summary'>
"""

```

OLS Regression Results

Dep. Variable:	CM	R-squared:	0.999
Model:	OLS	Adj. R-squared:	0.992
Method:	Least Squares	F-statistic:	146.2
Date:	Wed, 16 Aug 2023	Prob (F-statistic):	0.00682
Time:	19:16:43	Log-Likelihood:	62.359
No. Observations:	19	AIC:	-90.72
Df Residuals:	2	BIC:	-74.66
Df Model:	16		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
Intercept	1.7537	0.016	108.457	0.000	1.684	1.823
U	0.2881	0.007	41.152	0.001	0.258	0.318
A	-0.0364	0.007	-5.195	0.035	-0.066	-0.006
U:A	-0.0933	0.007	-13.319	0.006	-0.123	-0.063
P	0.0239	0.007	3.410	0.076	-0.006	0.054
U:P	0.0327	0.007	4.678	0.043	0.003	0.063

(continues on next page)

(continued from previous page)

```

A:P          -0.0610    0.007    -8.712    0.013    -0.091    -0.031
U:A:P        -0.0331    0.007    -4.731    0.042    -0.063    -0.003
Y            0.0271    0.007     3.874    0.061    -0.003     0.057
U:Y          0.0240    0.007     3.428    0.076    -0.006     0.054
A:Y          0.0027    0.007     0.393    0.732    -0.027     0.033
U:A:Y        -0.0819    0.007   -11.694    0.007    -0.112    -0.052
P:Y          -0.0210    0.007    -2.999    0.096    -0.051     0.009
U:P:Y        0.0231    0.007     3.303    0.081    -0.007     0.053
A:P:Y        0.0491    0.007     7.016    0.020     0.019     0.079
U:A:P:Y      0.0472    0.007     6.749    0.021     0.017     0.077
I(U ** 2)    -0.0243    0.004    -5.515    0.031    -0.043    -0.005
I(A ** 2)    -0.0243    0.004    -5.515    0.031    -0.043    -0.005
I(P ** 2)    -0.0243    0.004    -5.515    0.031    -0.043    -0.005
I(Y ** 2)    -0.0243    0.004    -5.515    0.031    -0.043    -0.005
=====
Omnibus:                13.078    Durbin-Watson:                3.023
Prob(Omnibus):          0.001    Jarque-Bera (JB):            33.486
Skew:                   0.110    Prob(JB):                    5.35e-08
Kurtosis:               9.500    Cond. No.                     1.96e+30
=====

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The input rank is higher than the number of observations.

[3] The smallest eigenvalue is 2.1e-59. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

"""

'Bso': <class 'statsmodels.iolib.summary.Summary'>

"""

OLS Regression Results

```

=====
Dep. Variable:          B    R-squared:                1.000
Model:                  OLS    Adj. R-squared:            0.999
Method:                 Least Squares    F-statistic:          1291.
Date:                   Wed, 16 Aug 2023    Prob (F-statistic):    0.000774
Time:                   19:16:43    Log-Likelihood:        -29.439
No. Observations:      19    AIC:                   92.88
Df Residuals:          2    BIC:                   108.9
Df Model:               16
Covariance Type:        nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	222.6667	2.028	109.819	0.000	213.943	231.391
U	-7.3125	0.878	-8.329	0.014	-11.090	-3.535
A	78.1875	0.878	89.055	0.000	74.410	81.965
U:A	-24.8125	0.878	-28.261	0.001	-28.590	-21.035
P	-10.8125	0.878	-12.315	0.007	-14.590	-7.035
U:P	8.4375	0.878	9.610	0.011	4.660	12.215
A:P	-25.8125	0.878	-29.400	0.001	-29.590	-22.035
U:A:P	-20.5625	0.878	-23.420	0.002	-24.340	-16.785

(continues on next page)

(continued from previous page)

Y	-64.9375	0.878	-73.963	0.000	-68.715	-61.160
U:Y	0.0625	0.878	0.071	0.950	-3.715	3.840
A:Y	-47.4375	0.878	-54.031	0.000	-51.215	-43.660
U:A:Y	-17.4375	0.878	-19.861	0.003	-21.215	-13.660
P:Y	19.5625	0.878	22.281	0.002	15.785	23.340
U:P:Y	13.3125	0.878	15.163	0.004	9.535	17.090
A:P:Y	21.5625	0.878	24.559	0.002	17.785	25.340
U:A:P:Y	8.3125	0.878	9.468	0.011	4.535	12.090
I(U ** 2)	0.2552	0.552	0.462	0.689	-2.121	2.632
I(A ** 2)	0.2552	0.552	0.462	0.689	-2.121	2.632
I(P ** 2)	0.2552	0.552	0.462	0.689	-2.121	2.632
I(Y ** 2)	0.2552	0.552	0.462	0.689	-2.121	2.632

Omnibus:	13.909	Durbin-Watson:	2.640
Prob(Omnibus):	0.001	Jarque-Bera (JB):	34.047
Skew:	-0.435	Prob(JB):	4.04e-08
Kurtosis:	9.500	Cond. No.	1.96e+30

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The input rank is higher than the number of observations.

[3] The smallest eigenvalue is 2.1e-59. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

''''',

'F': <class 'statsmodels.iolib.summary.Summary'>

''''''

OLS Regression Results

Dep. Variable:	F	R-squared:	0.992
Model:	OLS	Adj. R-squared:	0.954
Method:	Least Squares	F-statistic:	25.70
Date:	Wed, 16 Aug 2023	Prob (F-statistic):	0.0107
Time:	19:16:43	Log-Likelihood:	-49.091
No. Observations:	19	AIC:	130.2
Df Residuals:	3	BIC:	145.3
Df Model:	15		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
Intercept	78.5789	1.851	42.463	0.000	72.690	84.468
U	27.0625	2.017	13.420	0.001	20.645	33.480
A	-8.4375	2.017	-4.184	0.025	-14.855	-2.020
U:A	-8.0625	2.017	-3.998	0.028	-14.480	-1.645
P	9.6875	2.017	4.804	0.017	3.270	16.105
U:P	-2.9375	2.017	-1.457	0.241	-9.355	3.480
A:P	-4.6875	2.017	-2.324	0.103	-11.105	1.730
U:A:P	5.6875	2.017	2.820	0.067	-0.730	12.105
Y	12.5625	2.017	6.230	0.008	6.145	18.980
U:Y	4.6875	2.017	2.324	0.103	-1.730	11.105

(continues on next page)

(continued from previous page)

```

A:Y      -11.3125    2.017    -5.610    0.011    -17.730    -4.895
U:A:Y     -8.1875    2.017    -4.060    0.027    -14.605    -1.770
P:Y       4.8125    2.017     2.386    0.097     -1.605    11.230
U:P:Y     -7.5625    2.017    -3.750    0.033    -13.980    -1.145
A:P:Y     -9.3125    2.017    -4.618    0.019    -15.730    -2.895
U:A:P:Y   -0.6875    2.017    -0.341    0.756     -7.105     5.730

```

```

=====
Omnibus:                14.090    Durbin-Watson:                2.047
Prob(Omnibus):          0.001    Jarque-Bera (JB):           27.071
Skew:                   0.726    Prob(JB):                   1.32e-06
Kurtosis:               8.664    Cond. No.                   1.09
=====

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```

"""
'CM': <class 'statsmodels.iolib.summary.Summary'>
"""

```

OLS Regression Results

```

=====
Dep. Variable:          CM    R-squared:                0.986
Model:                  OLS    Adj. R-squared:           0.917
Method:                 Least Squares    F-statistic:              14.24
Date:                   Wed, 16 Aug 2023    Prob (F-statistic):       0.0250
Time:                   19:16:43    Log-Likelihood:           35.898
No. Observations:       19    AIC:                      -39.80
Df Residuals:           3    BIC:                      -24.69
Df Model:               15
Covariance Type:        nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	1.6718	0.021	79.166	0.000	1.605	1.739
U	0.2881	0.023	12.520	0.001	0.215	0.361
A	-0.0364	0.023	-1.581	0.212	-0.110	0.037
U:A	-0.0932	0.023	-4.052	0.027	-0.166	-0.020
P	0.0239	0.023	1.037	0.376	-0.049	0.097
U:P	0.0328	0.023	1.423	0.250	-0.040	0.106
A:P	-0.0610	0.023	-2.651	0.077	-0.134	0.012
U:A:P	-0.0331	0.023	-1.439	0.246	-0.106	0.040
Y	0.0271	0.023	1.179	0.323	-0.046	0.100
U:Y	0.0240	0.023	1.043	0.374	-0.049	0.097
A:Y	0.0028	0.023	0.119	0.912	-0.070	0.076
U:A:Y	-0.0819	0.023	-3.558	0.038	-0.155	-0.009
P:Y	-0.0210	0.023	-0.913	0.429	-0.094	0.052
U:P:Y	0.0231	0.023	1.005	0.389	-0.050	0.096
A:P:Y	0.0491	0.023	2.135	0.122	-0.024	0.122
U:A:P:Y	0.0473	0.023	2.053	0.132	-0.026	0.120

```

=====
Omnibus:                19.927    Durbin-Watson:                0.346
Prob(Omnibus):          0.000    Jarque-Bera (JB):           20.834

```

(continues on next page)

(continued from previous page)

```
Skew:                2.121    Prob(JB):                2.99e-05
Kurtosis:            5.884    Cond. No.                1.09
```

```
Notes:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is correctly
    specified.
```

```
====,
'B': <class 'statsmodels.iolib.summary.Summary'>
====
```

OLS Regression Results

```
=====
Dep. Variable:                B    R-squared:                1.000
Model:                        OLS    Adj. R-squared:            0.999
Method:                        Least Squares    F-statistic:            1867.
Date:                          Wed, 16 Aug 2023    Prob (F-statistic):      1.80e-05
Time:                          19:16:43    Log-Likelihood:         -30.403
No. Observations:              19    AIC:                    92.81
Df Residuals:                  3    BIC:                    107.9
Df Model:                      15
Covariance Type:                nonrobust
=====
```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	223.5263	0.692	322.991	0.000	221.324	225.729
U	-7.3125	0.754	-9.696	0.002	-9.713	-4.912
A	78.1875	0.754	103.677	0.000	75.787	80.588
U:A	-24.8125	0.754	-32.901	0.000	-27.213	-22.412
P	-10.8125	0.754	-14.337	0.001	-13.213	-8.412
U:P	8.4375	0.754	11.188	0.002	6.037	10.838
A:P	-25.8125	0.754	-34.227	0.000	-28.213	-23.412
U:A:P	-20.5625	0.754	-27.266	0.000	-22.963	-18.162
Y	-64.9375	0.754	-86.107	0.000	-67.338	-62.537
U:Y	0.0625	0.754	0.083	0.939	-2.338	2.463
A:Y	-47.4375	0.754	-62.902	0.000	-49.838	-45.037
U:A:Y	-17.4375	0.754	-23.122	0.000	-19.838	-15.037
P:Y	19.5625	0.754	25.940	0.000	17.162	21.963
U:P:Y	13.3125	0.754	17.652	0.000	10.912	15.713
A:P:Y	21.5625	0.754	28.592	0.000	19.162	23.963
U:A:P:Y	8.3125	0.754	11.022	0.002	5.912	10.713

```
=====
Omnibus:                  31.293    Durbin-Watson:                2.398
Prob(Omnibus):              0.000    Jarque-Bera (JB):              77.186
Skew:                      -2.374    Prob(JB):                      1.73e-17
Kurtosis:                   11.658    Cond. No.                      1.09
=====
```

```
Notes:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is correctly
    specified.
```

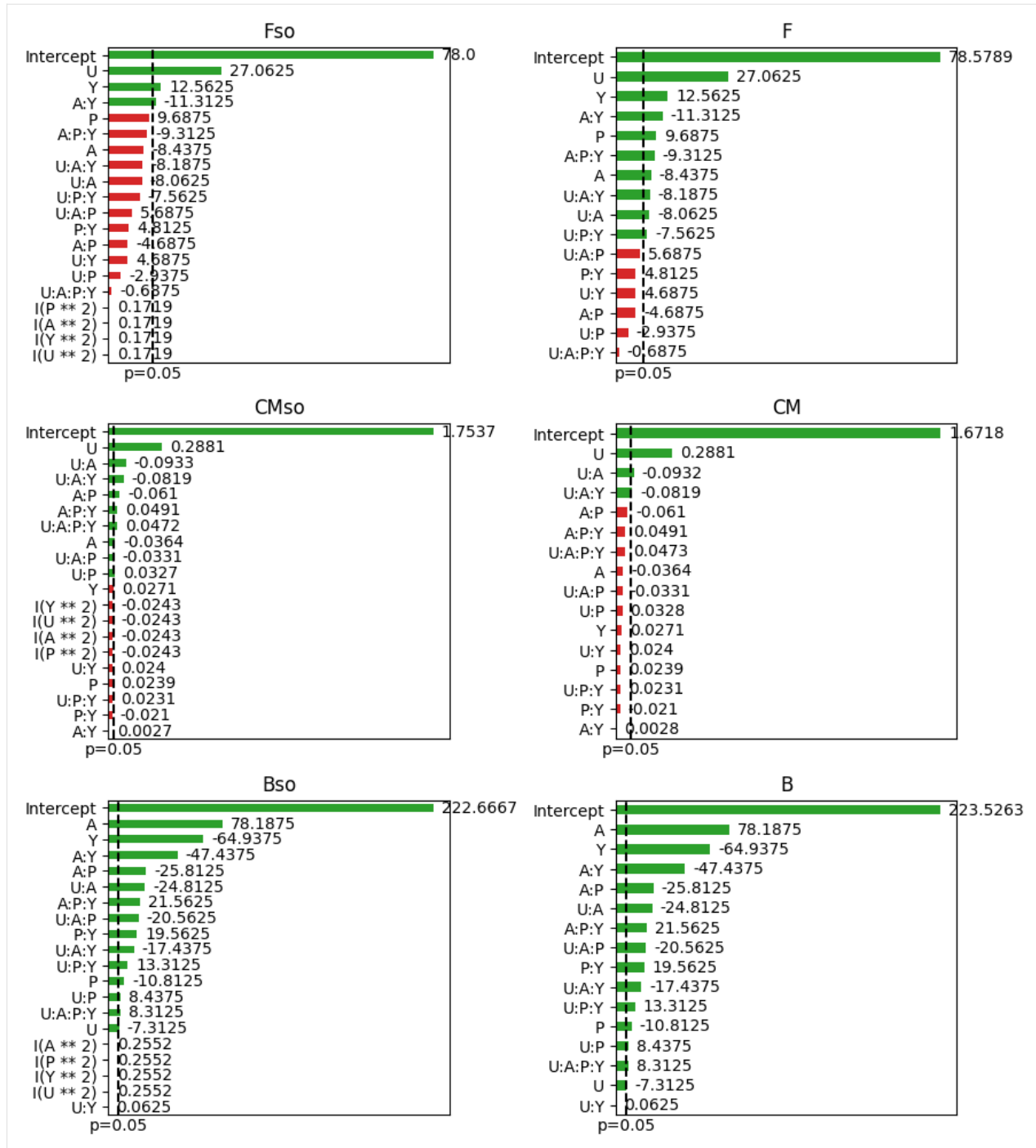
```
====}
```



```
[12]: from explann.plot import ParetoPlot

import matplotlib.pyplot as plt
fig, ax = plt.subplots(3,2, figsize=(9,10))
ax = ax.flatten()
pp_fm2b4 = ParetoPlot(fm2b4)

pp_fm2b4.plot(['Fso', 'F', 'CMso', 'CM', 'Bso', 'B'], ax=ax)
plt.tight_layout()
```



```
[13]: fm2b4.print_equation()
```

```
[13]: {'Fso': 'F = 78.0000 + 27.0625 * U + 12.5625 * Y - 11.3125 * AY',
      'CMso': 'CM = 1.7537 + 0.2881 * U - 0.0364 * A - 0.0933 * UA + 0.0327 * UP - 0.0610 * AP - 0.0331 * UAP - 0.0819 * UAY + 0.0491 * APY + 0.0472 * UAPY - 0.0243 * I(U ** 2) - 0.0243 * I(A ** 2) - 0.0243 * I(P ** 2) - 0.0243 * I(Y ** 2)',
      'Bso': 'B = 222.6667 - 7.3125 * U + 78.1875 * A - 24.8125 * UA - 10.8125 * P + 8.4375 * UP - 25.8125 * AP - 20.5625 * UAP - 64.9375 * Y - 47.4375 * AY - 17.4375 * UAY + 19.5625 * P:Y + 13.3125 * U:P:Y - 0.0625 * U:Y'}
```

(continues on next page)

(continued from previous page)

```

→5625 * PY + 13.3125 * UPY + 21.5625 * APY + 8.3125 * UAPY',
'F': 'F = 78.5789 + 27.0625 * U - 8.4375 * A - 8.0625 * UA + 9.6875 * P + 12.5625 * Y -
→11.3125 * AY - 8.1875 * UAY - 7.5625 * UPY - 9.3125 * APY',
'CM': 'CM = 1.6718 + 0.2881 * U - 0.0932 * UA - 0.0819 * UAY',
'B': 'B = 223.5263 - 7.3125 * U + 78.1875 * A - 24.8125 * UA - 10.8125 * P + 8.4375 *
→UP - 25.8125 * AP - 20.5625 * UAP - 64.9375 * Y - 47.4375 * AY - 17.4375 * UAY + 19.
→5625 * PY + 13.3125 * UPY + 21.5625 * APY + 8.3125 * UAPY'}

```

`build_significant_models` get the models with only significant terms withing a significance value α .

```
[14]: sig_fm2b4=fm2b4.build_significant_models(alpha=0.05)
sig_fm2b4.summary()
```

```

/home/ppiper/micromamba/envs/explann/lib/python3.9/site-packages/scipy/stats/_stats_py.
→py:1806: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=19
warnings.warn("kurtosistest only valid for n>=20 ... continuing ")
/home/ppiper/micromamba/envs/explann/lib/python3.9/site-packages/scipy/stats/_stats_py.
→py:1806: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=19
warnings.warn("kurtosistest only valid for n>=20 ... continuing ")
/home/ppiper/micromamba/envs/explann/lib/python3.9/site-packages/scipy/stats/_stats_py.
→py:1806: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=19
warnings.warn("kurtosistest only valid for n>=20 ... continuing ")
/home/ppiper/micromamba/envs/explann/lib/python3.9/site-packages/scipy/stats/_stats_py.
→py:1806: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=19
warnings.warn("kurtosistest only valid for n>=20 ... continuing ")
/home/ppiper/micromamba/envs/explann/lib/python3.9/site-packages/scipy/stats/_stats_py.
→py:1806: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=19
warnings.warn("kurtosistest only valid for n>=20 ... continuing ")

```

```
[14]: {'Fso': <class 'statsmodels.iolib.summary.Summary'>
      """
```

OLS Regression Results

```

=====
Dep. Variable:          F    R-squared:          0.644
Model:                OLS    Adj. R-squared:      0.573
Method:             Least Squares    F-statistic:      9.063
Date:                Wed, 16 Aug 2023    Prob (F-statistic): 0.00115
Time:                19:16:45    Log-Likelihood:    -85.472
No. Observations:      19    AIC:              178.9
Df Residuals:          15    BIC:              182.7
Df Model:                3
Covariance Type:      nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	78.5789	5.616	13.993	0.000	66.609	90.549
U	27.0625	6.120	4.422	0.000	14.019	40.106
Y	12.5625	6.120	2.053	0.058	-0.481	25.606
A:Y	-11.3125	6.120	-1.849	0.084	-24.356	1.731

```

=====

```

(continues on next page)

(continued from previous page)

```

Omnibus:                3.592    Durbin-Watson:                2.130
Prob(Omnibus):           0.166    Jarque-Bera (JB):        1.637
Skew:                   -0.592    Prob(JB):                0.441
Kurtosis:                3.815    Cond. No.                1.09
=====

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```

"""
'CMso': <class 'statsmodels.iolib.summary.Summary'>
"""

```

OLS Regression Results

```

=====
Dep. Variable:            CM    R-squared:                0.974
Model:                    OLS    Adj. R-squared:           0.942
Method:                    Least Squares    F-statistic:              30.17
Date:                      Wed, 16 Aug 2023    Prob (F-statistic):       2.87e-05
Time:                      19:16:45    Log-Likelihood:           29.977
No. Observations:          19    AIC:                     -37.95
Df Residuals:              8    BIC:                     -27.57
Df Model:                  10
Covariance Type:           nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	1.7537	0.044	39.456	0.000	1.651	1.856
U	0.2881	0.019	14.971	0.000	0.244	0.333
A	-0.0364	0.019	-1.890	0.095	-0.081	0.008
U:A	-0.0933	0.019	-4.845	0.001	-0.138	-0.049
U:P	0.0328	0.019	1.702	0.127	-0.012	0.077
A:P	-0.0610	0.019	-3.170	0.013	-0.105	-0.017
U:A:P	-0.0331	0.019	-1.721	0.124	-0.078	0.011
U:A:Y	-0.0819	0.019	-4.254	0.003	-0.126	-0.037
A:P:Y	0.0491	0.019	2.553	0.034	0.005	0.094
U:A:P:Y	0.0473	0.019	2.455	0.040	0.003	0.092
I(U ** 2)	-0.0243	0.012	-2.006	0.080	-0.052	0.004
I(A ** 2)	-0.0243	0.012	-2.006	0.080	-0.052	0.004
I(P ** 2)	-0.0243	0.012	-2.006	0.080	-0.052	0.004
I(Y ** 2)	-0.0243	0.012	-2.006	0.080	-0.052	0.004

```

=====
Omnibus:                2.073    Durbin-Watson:                1.950
Prob(Omnibus):           0.355    Jarque-Bera (JB):        1.028
Skew:                   -0.013    Prob(JB):                0.598
Kurtosis:                1.861    Cond. No.                2.87e+16
=====

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The smallest eigenvalue is 9.82e-32. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

(continues on next page)

(continued from previous page)

```

"""
'Bso': <class 'statsmodels.iolib.summary.Summary'>
"""

```

OLS Regression Results

```

=====
Dep. Variable:          B    R-squared:          1.000
Model:                  OLS    Adj. R-squared:    1.000
Method:                 Least Squares    F-statistic:    2661.
Date:                   Wed, 16 Aug 2023    Prob (F-statistic):    3.23e-07
Time:                   19:16:45    Log-Likelihood:    -30.425
No. Observations:      19    AIC:          90.85
Df Residuals:          4    BIC:          105.0
Df Model:              14
Covariance Type:       nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	223.5263	0.600	372.531	0.000	221.860	225.192
U	-7.3125	0.654	-11.184	0.000	-9.128	-5.497
A	78.1875	0.654	119.579	0.000	76.372	80.003
U:A	-24.8125	0.654	-37.948	0.000	-26.628	-22.997
P	-10.8125	0.654	-16.536	0.000	-12.628	-8.997
U:P	8.4375	0.654	12.904	0.000	6.622	10.253
A:P	-25.8125	0.654	-39.477	0.000	-27.628	-23.997
U:A:P	-20.5625	0.654	-31.448	0.000	-22.378	-18.747
Y	-64.9375	0.654	-99.315	0.000	-66.753	-63.122
A:Y	-47.4375	0.654	-72.550	0.000	-49.253	-45.622
U:A:Y	-17.4375	0.654	-26.669	0.000	-19.253	-15.622
P:Y	19.5625	0.654	29.919	0.000	17.747	21.378
U:P:Y	13.3125	0.654	20.360	0.000	11.497	15.128
A:P:Y	21.5625	0.654	32.977	0.000	19.747	23.378
U:A:P:Y	8.3125	0.654	12.713	0.000	6.497	10.128

```

=====
Omnibus:                 31.153    Durbin-Watson:          2.404
Prob(Omnibus):           0.000    Jarque-Bera (JB):        76.327
Skew:                    -2.365    Prob(JB):                2.67e-17
Kurtosis:                11.605    Cond. No.                1.09
=====

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```

"""
'F': <class 'statsmodels.iolib.summary.Summary'>
"""

```

OLS Regression Results

```

=====
Dep. Variable:          F    R-squared:          0.924
Model:                  OLS    Adj. R-squared:    0.847
Method:                 Least Squares    F-statistic:    12.08
Date:                   Wed, 16 Aug 2023    Prob (F-statistic):    0.000499
Time:                   19:16:45    Log-Likelihood:    -70.868
=====

```

(continues on next page)

(continued from previous page)

```

No. Observations:      19  AIC:      161.7
Df Residuals:          9  BIC:      171.2
Df Model:              9
Covariance Type:      nonrobust

```

```

=====
              coef      std err          t      P>|t|      [0.025      0.975]
-----
Intercept    78.5789      3.361     23.377     0.000     70.975     86.183
U             27.0625      3.663      7.388     0.000     18.776     35.349
A            -8.4375      3.663     -2.303     0.047    -16.724     -0.151
U:A          -8.0625      3.663     -2.201     0.055    -16.349      0.224
P             9.6875      3.663      2.645     0.027      1.401     17.974
Y            12.5625      3.663      3.430     0.008      4.276     20.849
A:Y          -11.3125      3.663     -3.088     0.013    -19.599     -3.026
U:A:Y        -8.1875      3.663     -2.235     0.052    -16.474      0.099
U:P:Y        -7.5625      3.663     -2.065     0.069    -15.849      0.724
A:P:Y        -9.3125      3.663     -2.542     0.032    -17.599     -1.026
=====
Omnibus:            0.222  Durbin-Watson:      1.718
Prob(Omnibus):      0.895  Jarque-Bera (JB):      0.374
Skew:              -0.200  Prob(JB):            0.830
Kurtosis:          2.441  Cond. No.            1.09
=====

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```

"""
'CM': <class 'statsmodels.iolib.summary.Summary'>
"""

```

OLS Regression Results

```

=====
Dep. Variable:      CM  R-squared:      0.858
Model:              OLS  Adj. R-squared:  0.829
Method:             Least Squares  F-statistic:    30.16
Date:               Wed, 16 Aug 2023  Prob (F-statistic): 1.34e-06
Time:               19:16:45  Log-Likelihood: 13.772
No. Observations:   19  AIC:            -19.54
Df Residuals:       15  BIC:            -15.77
Df Model:           3
Covariance Type:    nonrobust
=====
              coef      std err          t      P>|t|      [0.025      0.975]
-----
Intercept    1.6718      0.030     55.244     0.000      1.607      1.736
U             0.2881      0.033      8.737     0.000      0.218      0.358
U:A          -0.0933      0.033     -2.828     0.013     -0.164     -0.023
U:A:Y        -0.0819      0.033     -2.483     0.025     -0.152     -0.012
=====
Omnibus:            3.181  Durbin-Watson:      1.725
Prob(Omnibus):      0.204  Jarque-Bera (JB):      1.658
Skew:              -0.704  Prob(JB):            0.437

```

(continues on next page)

(continued from previous page)

```
Kurtosis:                3.336    Cond. No.                1.09
```

```
Notes:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is correctly
```

```
↪specified.
```

```
""",
```

```
'B': <class 'statsmodels.iolib.summary.Summary'>
```

```
"""
```

OLS Regression Results

```
=====
Dep. Variable:                B    R-squared:                1.000
Model:                        OLS    Adj. R-squared:        1.000
Method:                        Least Squares    F-statistic:        2661.
Date:                        Wed, 16 Aug 2023    Prob (F-statistic):    3.23e-07
Time:                        19:16:45    Log-Likelihood:        -30.425
No. Observations:            19    AIC:                90.85
Df Residuals:                4    BIC:                105.0
Df Model:                    14
Covariance Type:            nonrobust
=====
```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	223.5263	0.600	372.531	0.000	221.860	225.192
U	-7.3125	0.654	-11.184	0.000	-9.128	-5.497
A	78.1875	0.654	119.579	0.000	76.372	80.003
U:A	-24.8125	0.654	-37.948	0.000	-26.628	-22.997
P	-10.8125	0.654	-16.536	0.000	-12.628	-8.997
U:P	8.4375	0.654	12.904	0.000	6.622	10.253
A:P	-25.8125	0.654	-39.477	0.000	-27.628	-23.997
U:A:P	-20.5625	0.654	-31.448	0.000	-22.378	-18.747
Y	-64.9375	0.654	-99.315	0.000	-66.753	-63.122
A:Y	-47.4375	0.654	-72.550	0.000	-49.253	-45.622
U:A:Y	-17.4375	0.654	-26.669	0.000	-19.253	-15.622
P:Y	19.5625	0.654	29.919	0.000	17.747	21.378
U:P:Y	13.3125	0.654	20.360	0.000	11.497	15.128
A:P:Y	21.5625	0.654	32.977	0.000	19.747	23.378
U:A:P:Y	8.3125	0.654	12.713	0.000	6.497	10.128

```
=====
Omnibus:                    31.153    Durbin-Watson:        2.404
Prob(Omnibus):              0.000    Jarque-Bera (JB):      76.327
Skew:                       -2.365    Prob(JB):              2.67e-17
Kurtosis:                   11.605    Cond. No.              1.09
=====
```

```
Notes:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is correctly
```

```
↪specified.
```

```
"""}

```

```
[15]: sig_fm2b4.anova('F')
```

[15]:

	df	sum_sq	mean_sq	F	PR(>F)
U	1.0	11718.062500	11718.062500	54.585296	0.000042
A	1.0	1139.062500	1139.062500	5.306002	0.046733
U:A	1.0	1040.062500	1040.062500	4.844838	0.055241
P	1.0	1501.562500	1501.562500	6.994606	0.026706
Y	1.0	2525.062500	2525.062500	11.762293	0.007513
A:Y	1.0	2047.562500	2047.562500	9.537994	0.012964
U:A:Y	1.0	1072.562500	1072.562500	4.996231	0.052249
U:P:Y	1.0	915.062500	915.062500	4.262561	0.068965
A:P:Y	1.0	1387.562500	1387.562500	6.463569	0.031588
Residual	9.0	1932.069079	214.674342	NaN	NaN

2.2.5 Factorial 2^3

Factor	Min −1	CP 0	Max +1
Urea (g/L)	0.30	0.55	0.80
Ammonium sulfate (g/L)	1.20	2.60	4.00
Yeast extract (g/L)	0.08	0.16	0.25
<i>Min</i> minimum value, <i>CP</i> center point, <i>Max</i> maximum value.			
Levels for factorial 2^3 experimental design.			

Run	Nitrogen sources			Activities (U/L)		
	Urea	Ammonium sulfate	Yeast extract	FPase	CMCase	β-glucosidase
1	−1	−1	−1	158	3,995	835
2	+1	−1	−1	202	4,291	1,540
3	−1	+1	−1	137	3,642	1,021
4	+1	+1	−1	240	4,895	1,702
5	−1	−1	+1	191	4,311	1,311
6	+1	−1	+1	221	4,520	1,717
7	−1	+1	+1	141	4,726	1,345
8	+1	+1	+1	175	5,144	1,728
9 (CP)	0	0	0	225	4,715	1,623
10 (CP)	0	0	0	234	4,565	1,687
11 (CP)	0	0	0	250	4,743	1,713
<i>CP</i> center point						

Experimental results for factorial 2^3

2.2.6 building a model with explann

```
[16]: from explann.doe import TwoLevelFactorial
from explann.models import FactorialModel
from explann.dataio import ImportString, ImportXLSX
from explann.plot import ParetoPlot
import matplotlib.pyplot as plt

f2b3 = TwoLevelFactorial(
    variables = {
        'U': (0.3, 0.8),
        'A': (1.20, 4.00),
        'Y': (0.08, 0.25)
    },
    central_points=3
)

f2b3_results = ImportString(
    data=
    """F,CM,B
158,3995,835
202,4291,1540
137,3642,1021
240,4895,1702
191,4311,1311
221,4520,1717
141,4726,1345
175,5144,1728
225,4715,1623
234,4565,1687
250,4743,1713
""",
    delimiter=',')

f2b3.append_results(results=f2b3_results.data)

f2b3.save_excel('../data/f2b3.xlsx')

f2b3_from_excel = ImportXLSX(
    path = '../data/f2b3.xlsx',
    levels_sheet = 'levels'
)

fm_f2b3 = FactorialModel(
    data = f2b3_from_excel.data,
    functions = {
        "Fso" : "F ~ U * A * Y + I(U**2) + I(A**2) + I(Y**2)",
        "CMso" : "CM ~ U * A * Y + I(U**2) + I(A**2) + I(Y**2)",
        "Bso" : "B ~ U * A * Y + I(U**2) + I(A**2) + I(Y**2)",
        "F" : "F ~ U * A * Y",
        "CM" : "CM ~ U * A * Y",
```

(continues on next page)

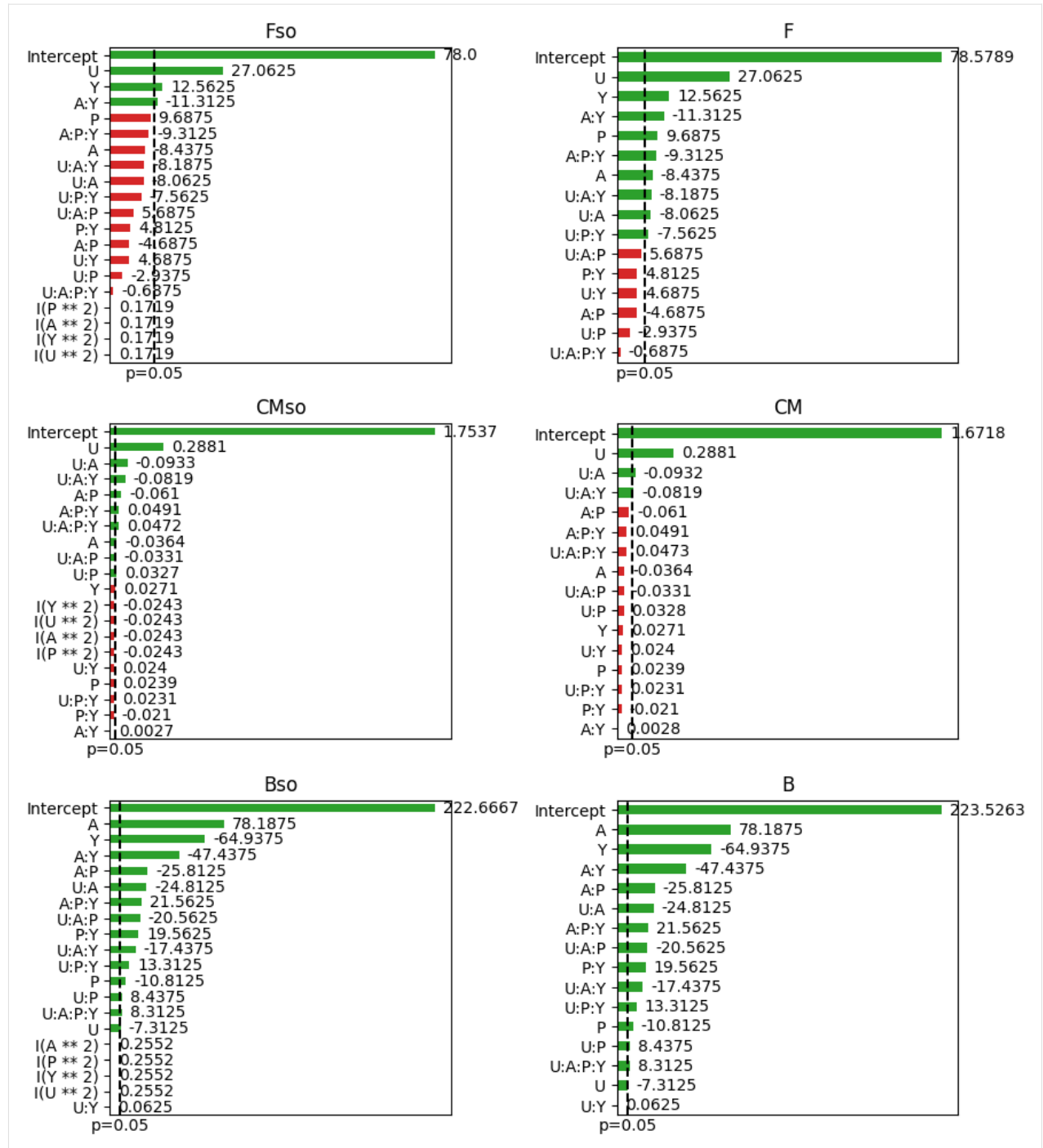
(continued from previous page)

```
    "B" : "B ~ U * A * Y"}
)

pp_f2b3 = ParetoPlot(fm_f2b3)

fig, ax = plt.subplots(3,2, figsize=(9,10))

pp_fm2b4.plot(['Fso', 'F', 'CMso', 'CM', 'Bso', 'B'], ax=ax)
plt.tight_layout()
```



2.2.7 Central Composite Design

Factor	Axial	Min	CP	Max	Axial
	-1.41	-1	0	+1	+1.41
Urea (g/L)	0.07	0.40	1.20	2.00	2.33
Yeast extract (g/L)	0.00	0.09	0.29	0.50	0.59

Min minimum value, *CP* center point, *Max* maximum value.

Levels for central composite experimental design.

Experiment	Nitrogen sources		Activities (U/L)		
	Urea	Yeast extract	FPase	CMCase	β -glucosidase
1	-1	-1	158	4,029	727
2	-1	+1	171	4,354	1,119
3	+1	-1	166	5,302	1,080
4	+1	+1	244	5,513	1,101
5	-1.41	0	148	4,481	743
6	+1.41	0	263	6,529	1,213
7	0	-1.41	208	5,460	1,085
8	0	+1.41	255	7,105	1,435
9 (CP)	0	0	250	5,364	1,390
10 (CP)	0	0	269	5,524	1,499
11 (CP)	0	0	261	5,793	1,420

CP center point

Experimental results for ccd.

```
[146]: from explann.doe import CentralCompositeDesign
from explann.models import FactorialModel
from explann.dataio import ImportString, ImportXLSX
from explann.plot import ParetoPlot
import matplotlib.pyplot as plt

ccd = CentralCompositeDesign(
    variables = {
        'U': (0.07, 2.33),
        'Y': (0.00, 0.59)
    },
    center=(0,3)
```

(continues on next page)

(continued from previous page)

```

)

ccd_results = ImportString(
    data=
    ""F,CM,B
    158,4029,727
    171,4354,1119
    166,5302,1080
    244,5513,1101
    208,5460,1085
    255,7105,1435
    148,4481,743
    263,6529,1213
    250,5364,1390
    269,5524,1499
    261,5793,1420
    "",
    delimiter=',')

ccd.append_results(results=ccd_results.data)

ccd.save_excel('../data/ccd.xlsx')

ccd_from_excel = ImportXLSX(
    path = '../data/ccd.xlsx',
    levels_sheet = 'levels'
)

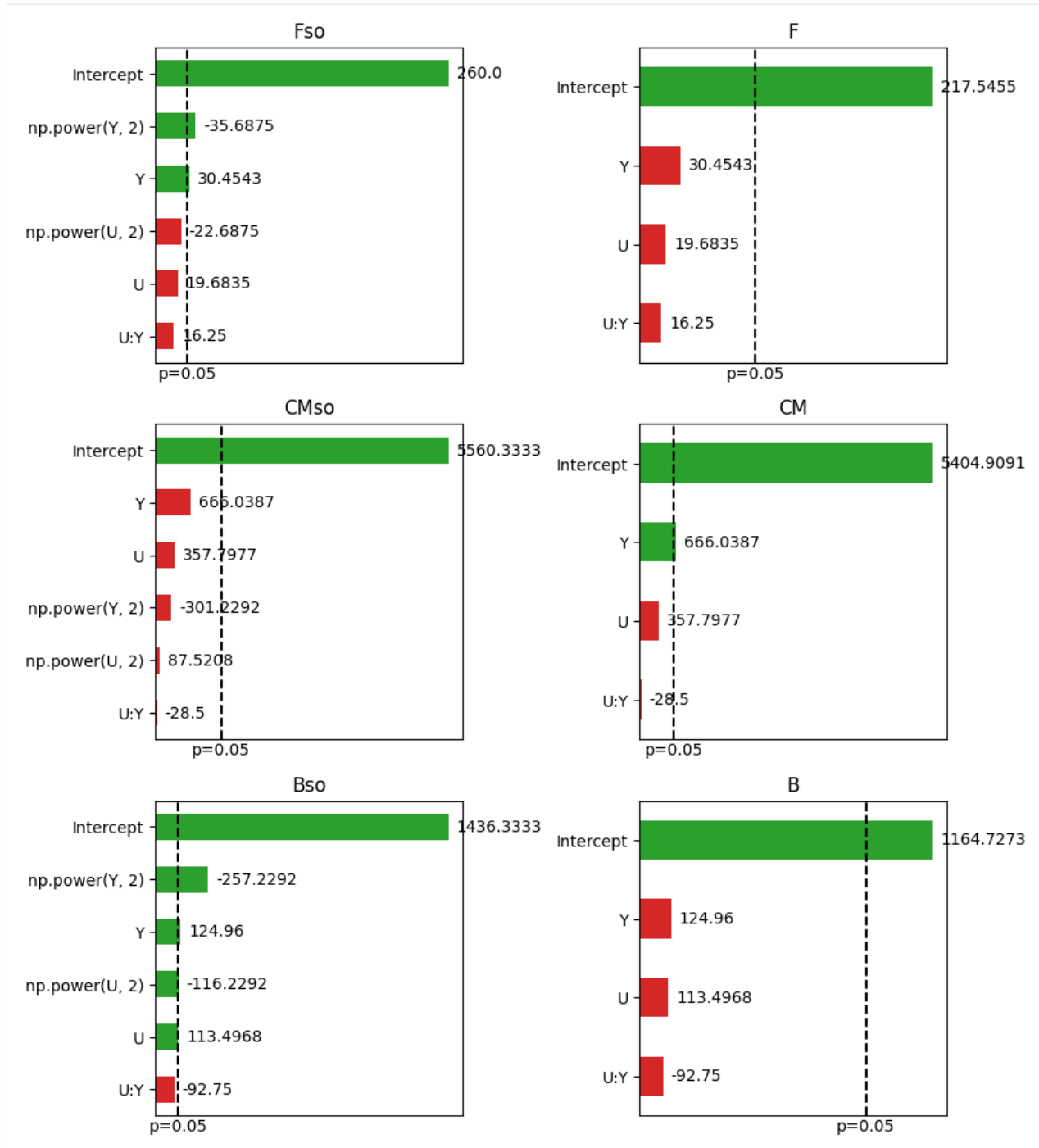
fm_ccd = FactorialModel(
    data = ccd_from_excel.data,
    functions = {
        # "Fso" : "F ~ U * Y + I(U**2) + I(Y**2)",
        # "CMso" : "CM ~ U * Y + I(U**2) + I(Y**2)",
        # "Bso" : "B ~ U * Y + I(U**2) + I(Y**2)",
        "Fso" : "F ~ U * Y + np.power(U,2) + np.power(Y,2)",
        "CMso" : "CM ~ U * Y + np.power(U,2) + np.power(Y,2)",
        "Bso" : "B ~ U * Y + np.power(U,2) + np.power(Y,2)",
        "F" : "F ~ U * Y",
        "CM" : "CM ~ U * Y",
        "B" : "B ~ U * Y"},
    levels = ccd_from_excel.levels,
)

pp_ccd = ParetoPlot(fm_ccd)

fig, ax = plt.subplots(3,2, figsize=(9,10))

pp_ccd.plot(['Fso', 'F', 'CMso', 'CM', 'Bso', 'B'], ax=ax)
plt.tight_layout()

```



```
[147]: import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from matplotlib import cm # for a scatter plot
from mpl_toolkits.mplot3d import Axes3D

def plot_surface(x, y, z, model, n_pts=10, other_params={}, labels=dict=None, ax=None,
    cmap='viridis', scaled=False):
```

(continues on next page)

(continued from previous page)

```

name_z = z #'Fso'
name_x = x #'U'
name_y = y #'Y'
n_pts = 10
if not scaled:
    X = np.linspace( model.data[name_x].min(), model.data[name_x].max(), n_pts)
    Y = np.linspace( model.data[name_y].min(), model.data[name_y].max(), n_pts)
else:
    X = np.linspace( model.levels[name_x].min(), model.levels[name_x].max(), n_pts)
    Y = np.linspace( model.levels[name_y].min(), model.levels[name_y].max(), n_pts)

x, y = np.meshgrid(X, Y)

try:
    variables = pd.DataFrame({name_x:x.ravel(), name_y:y.ravel(), **other_params})
except:
    variables = pd.DataFrame([name_x:x.ravel(), name_y:y.ravel(), **other_params])

if not scaled:
    z = model.predict(name_z, variables).values.reshape(x.shape)
else:
    z = model.predict_rescaled(name_z, variables).values.reshape(x.shape)

if ax is None:
    fig, ax = plt.subplots(figsize=(6,6),subplot_kw={"projection": "3d"})
else:
    fig = plt.gcf()
ax.set_box_aspect(None, zoom=0.8)

pax = ax.plot_surface(x, y, z, cmap=cmap, edgecolor='black', linewidth=0.5, alpha=0.
↪6, antialiased=True)
#ax.contour(x, y, z, cmap=cmap, linestyle='solid', alpha=1)
ax.contourf(x, y, z, zdir='z', offset=ax.get_zlim()[0], cmap=cmap, alpha=0.5, ↪
↪antialiased=True)
ax.contourf(x, y, z, zdir='x', offset=ax.get_xlim()[0], cmap=cmap, alpha=0.5, ↪
↪antialiased=True)
ax.contourf(x, y, z, zdir='y', offset=ax.get_ylim()[1], cmap=cmap, alpha=0.5, ↪
↪antialiased=True)

if labels is not None:
    ax.set(**labels)
    ax.zaxis.set_rotate_label(True)
    ax.xaxis.set_rotate_label(True)
    ax.yaxis.set_rotate_label(True)

try:
    if 'zlabel' in labels:
        zlabel=labels['zlabel']
except:
    zlabel=None

```

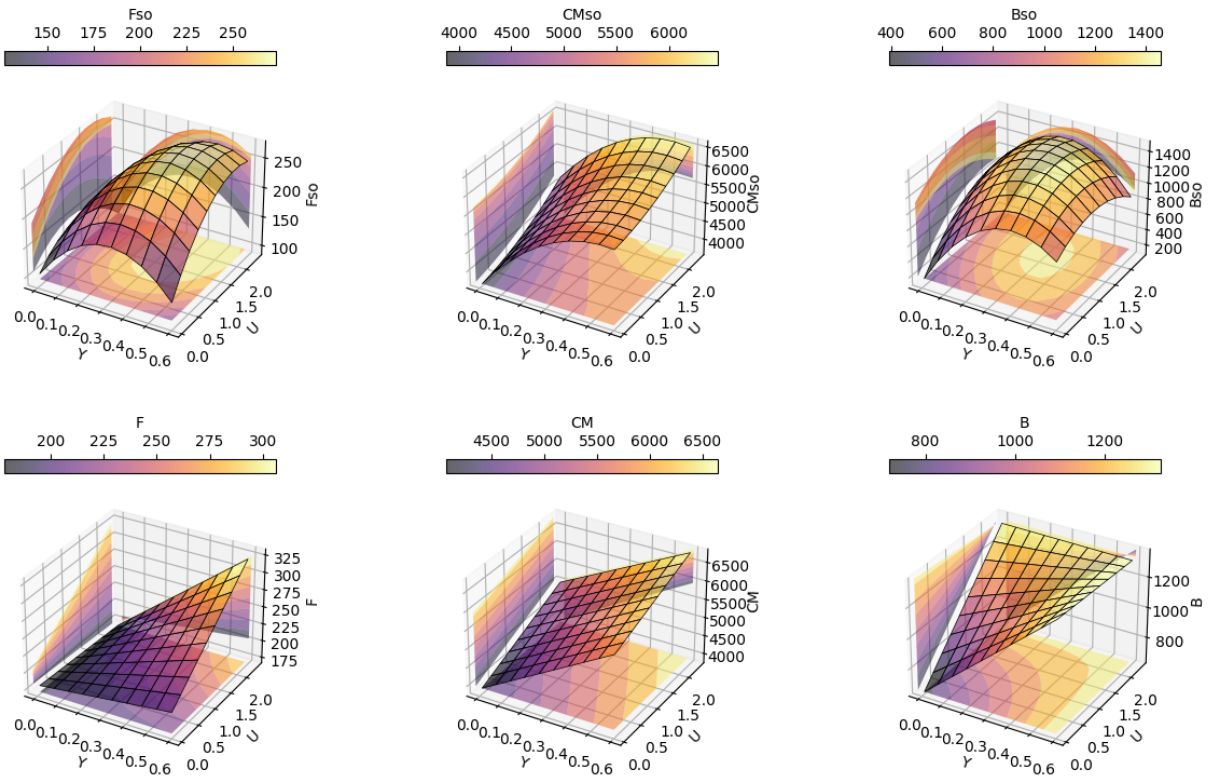
(continues on next page)

(continued from previous page)

```
fig.colorbar(pax, ax=ax, location='top', fraction=0.04, pad=-0.05, label=zlabel)
```

```
[148]: fig, ax = plt.subplots(2,3, figsize=(15,9),subplot_kw={"projection": "3d"})
ax = ax.flatten()

for i,(key,val) in enumerate(fm_ccd.functions.items()):
    plot_surface(x='Y', y='U', z=key,
                model=fm_ccd,
                n_pts=10,
                ax=ax[i],
                other_params=dict(),
                labels=dict(xlabel='Y', ylabel='U', zlabel=key), cmap='inferno', scaled=True)
```



2.2.8 Optimization Using Desirability Function

A desirability function D is used to optimize the two independent variables at the same time.

$$D = \left(\sum_{i=1}^N d_i^N \right)^{(1/N)}$$

The function d_i was defined as

```
[285]: @latexify.with_latex
def d_i(Y, Ymin=Ymin, Ymax=Ymax, r=1):
    if Y <= Ymin:
```

(continues on next page)

(continued from previous page)

```

    return 0
elif Y > Ymin and Y < Ymax:
    return ((Y-Ymin)/(Ymax-Ymin))**r
else:
    return 1

```

d_i

[285]:

$$d_i(Y, Y_{min}, Y_{max}, r) = \begin{cases} 0, & \text{if } Y \leq Y_{min} \\ \left(\frac{Y - Y_{min}}{Y_{max} - Y_{min}} \right)^r, & \text{if } Y > Y_{min} \wedge Y < Y_{max} \\ 1, & \text{otherwise} \end{cases}$$

[269]:

```

from scipy.optimize import minimize
import latexify

responses = ['Fso', 'CMso', 'Bso']
model = fm_ccd
Ymax = 7000
Ymin = 200

def D(independent_vars):
    d_list = []
    for Yi in responses:
        d_list.append( d_i(model[Yi].predict(dict(U=independent_vars[0], Y=independent_
↪vars[1]))[0]) )
    d_array = np.array(d_list)
    n = len(d_array)
    return 1-(np.prod(d_array)**(1/n))

from scipy import optimize as opt

x0 = (0, 0)
bounds = [(-1.41, 1.41), (-1.41, 1.41)]

optimum = minimize(D, x0, method='SLSQP', bounds=bounds)
optimum

```

[269]:

```

message: Optimization terminated successfully
success: True
status: 0
  fun: 0.8802039525109903
   x: [ 6.344e-01  4.816e-01]
  nit: 10
  jac: [ 1.291e-04  2.973e-05]
 nfev: 30
 njev: 10

```

2.2.9 Anova table and lack of fit

To check model validity an ANOVA test and lack of fit can be used.

```
[270]: fm_ccd.anova('Fso')
```

```
[270]:
```

	df	sum_sq	mean_sq	F	PR(>F)
U	1.0	3099.522852	3099.522852	4.600914	0.084782
Y	1.0	7419.724833	7419.724833	11.013797	0.021038
U:Y	1.0	1056.250000	1056.250000	1.567891	0.265894
np.power(U, 2)	1.0	918.771390	918.771390	1.363819	0.295530
np.power(Y, 2)	1.0	7192.080882	7192.080882	10.675884	0.022263
Residual	5.0	3368.377315	673.675463	NaN	NaN

```
[271]: fm_ccd.lack_of_fit('Fso', alpha=0.05)
```

```
[271]:
```

	Source_of_Variation	df	sum_sq	mean_sq	F	F_table \
0	Regression	3.0	11575.497685	3858.499228	2.352901	4.346831
1	Residual	7.0	11479.229588	1639.889941	NaN	NaN
2	Lack_of_Fit	5.0	11297.229588	2259.445918	24.829076	19.296410
3	Pure_Error	2.0	182.000000	91.000000	NaN	NaN
4	Total	10.0	23054.727273	NaN	NaN	NaN

	p
0	0.158352
1	NaN
2	0.039167
3	NaN
4	NaN

2.2.10 Optimization Results

In the original papaer authors have found a desirability function of 0.87 and an optimum point at

$$U = 0.97 \text{ g/L}$$

$$Y = 0.36 \text{ g/L}$$

The corresponding cellulase activities of

$$F = 254 \text{ U/L}$$

$$C = 6383 \text{ U/L}$$

$$B = 1448 \text{ U/L}$$

Totaling a production of 8085 U/L. In this work the desirabilty function was 0.88 and the optimum point found at

$$U = 1.71 \text{ g/L}$$

$$Y = 0.40 \text{ g/L}$$

This correpsonts to a total cellulase activity of 8106 U/L, from which partial values for each metric are

$$F = 245 \text{ U/L}$$

$$C = 6623 \text{ U/L}$$

$$B = 1238 \text{ U/L}$$

```
[280]: fm_ccd.decode_variables(variables=dict(U=optimum.x[0], Y=optimum.x[1]))
```

```
[280]: {'U': 1.7069155936557219, 'Y': 0.39546634255213986}
```

```
[281]: F=fm_ccd.predict(function='Fso', variables=fm_ccd.decode_
↳variables(variables=dict(U=optimum.x[0], Y=optimum.x[1])))
F
```

```
[281]: 0    244.92823
dtype: float64
```

```
[282]: C=fm_ccd.predict(function='CMso', variables=fm_ccd.decode_
↳variables(variables=dict(U=optimum.x[0], Y=optimum.x[1])))
C
```

```
[282]: 0    6623.108285
dtype: float64
```

```
[283]: B=fm_ccd.predict(function='Bso', variables=fm_ccd.decode_
↳variables(variables=dict(U=optimum.x[0], Y=optimum.x[1])))
B
```

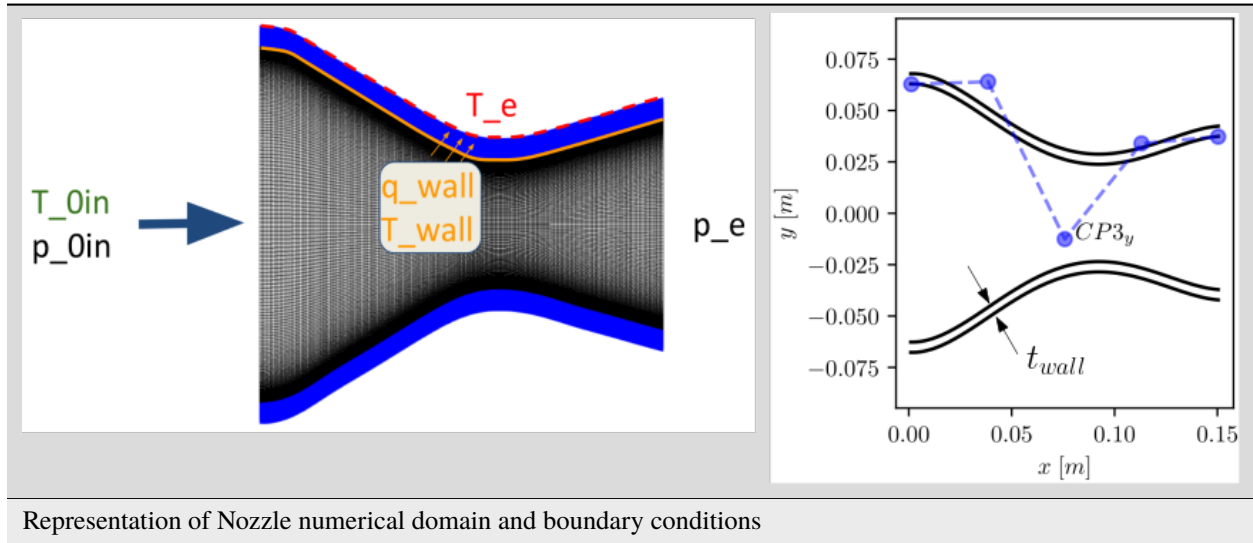
```
[283]: 0    1238.00179
dtype: float64
```

2.2.11 Proposal Study

This proposal employs a factorial design approach to construct a surrogate model. The aim is the reconstruction of 2D fields acquired through Reynolds Averaged Navier Stokes (RANS) solutions, involving conjugate heat transfer of the nozzle flow.

Nozzle Flow and Heat Transfer

Rocket nozzles hold paramount importance in aerospace propulsion. The optimal aerodynamic design of these nozzles is inherently limited by the need to maintain structural integrity, as they are subjected to intense heat transfer from the hot gases to the inner walls.



2.2.12 Main Hypotesis

The central hypothesis poses that the factorial design could be effective in acquiring a model with the capacity to predict the principal components of a reduced-order model of fluid flow and heat transfer. This model, once established, can then be used to reconstruct complete two-dimensional flow fields from the designated independent variables.

Experimental Samples

To test this hypothesis, a series of numerical experiments will be conducted following the Central Composite Design (CCD)

- Inlet Total Temperature, T_{0in} [K]
- Inlet Total Pressure, p_{0in} [Pa]
- Wall Thickness, t_{wall} [m]
- Nozzle Shape Control Point, $CP3_y$ [m]
- External Nozzle Wall Temperature, T_e [K]

The SU2 software (Economon et al., 2016) was employed to address the conjugate heat transfer interfaces between the fluid and solid in the nozzle. In the fluid domain, the Reynolds Averaged Navier-Stokes equations were solved using the finite volume method and the SST (Shear Stress Transport) turbulence model. To obtain the steady-state solution, the implicit Euler integration method was utilized in conjunction with time marching. On the other hand, for the solid domain, the energy equation was solved.

2.2.13 Central Composite Design

The range of variables were choosen in order to be within real experiments made by (Back et al., 1964) . Another factor taking into account to limit variable ranges is the numerical stability of conjugate heat transfer CFD.

```
[1]: from explann.doe import CentralCompositeDesign
import pandas as pd

variables = {
    't_wall': [0.001, 0.010],          #[m]
    'CP3_y': [-0.01255805, 0.0],       #[m]
    'T_0in': [400.0, 650.0],           #[K]
    'p_0in': [4.0e5, 1.0e6],           #[Pa]
    'T_e': [290.0, 400.0],             #[K]
}

# variable ranges
pd.DataFrame(variables)
```

```
[1]:   t_wall    CP3_y  T_0in    p_0in    T_e
0   0.001 -0.012558  400.0  400000.0  290.0
1   0.010  0.000000  650.0  1000000.0  400.0
```

A CCD (Central Composite Desig) was choosen in order to be able to capture any high order interaction fo the design variables.

```
[2]: ccd = CentralCompositeDesign(
    variables=variables,
    center=(1,0),
    alpha='r',
    face='ccc',
)
ccd.doe
```

```
[2]:
```

	t_wall	CP3_y	T_0in	p_0in	T_e
Index					
1	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
2	1.000000	-1.000000	-1.000000	-1.000000	-1.000000
3	-1.000000	1.000000	-1.000000	-1.000000	-1.000000
4	1.000000	1.000000	-1.000000	-1.000000	-1.000000
5	-1.000000	-1.000000	1.000000	-1.000000	-1.000000
6	1.000000	-1.000000	1.000000	-1.000000	-1.000000
7	-1.000000	1.000000	1.000000	-1.000000	-1.000000
8	1.000000	1.000000	1.000000	-1.000000	-1.000000
9	-1.000000	-1.000000	-1.000000	1.000000	-1.000000
10	1.000000	-1.000000	-1.000000	1.000000	-1.000000
11	-1.000000	1.000000	-1.000000	1.000000	-1.000000
12	1.000000	1.000000	-1.000000	1.000000	-1.000000
13	-1.000000	-1.000000	1.000000	1.000000	-1.000000
14	1.000000	-1.000000	1.000000	1.000000	-1.000000
15	-1.000000	1.000000	1.000000	1.000000	-1.000000
16	1.000000	1.000000	1.000000	1.000000	-1.000000
17	-1.000000	-1.000000	-1.000000	-1.000000	1.000000
18	1.000000	-1.000000	-1.000000	-1.000000	1.000000
19	-1.000000	1.000000	-1.000000	-1.000000	1.000000

(continues on next page)

(continued from previous page)

20	1.0000000	1.0000000	-1.0000000	-1.0000000	1.0000000
21	-1.0000000	-1.0000000	1.0000000	-1.0000000	1.0000000
22	1.0000000	-1.0000000	1.0000000	-1.0000000	1.0000000
23	-1.0000000	1.0000000	1.0000000	-1.0000000	1.0000000
24	1.0000000	1.0000000	1.0000000	-1.0000000	1.0000000
25	-1.0000000	-1.0000000	-1.0000000	1.0000000	1.0000000
26	1.0000000	-1.0000000	-1.0000000	1.0000000	1.0000000
27	-1.0000000	1.0000000	-1.0000000	1.0000000	1.0000000
28	1.0000000	1.0000000	-1.0000000	1.0000000	1.0000000
29	-1.0000000	-1.0000000	1.0000000	1.0000000	1.0000000
30	1.0000000	-1.0000000	1.0000000	1.0000000	1.0000000
31	-1.0000000	1.0000000	1.0000000	1.0000000	1.0000000
32	1.0000000	1.0000000	1.0000000	1.0000000	1.0000000
33	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
34	-2.378414	0.0000000	0.0000000	0.0000000	0.0000000
35	2.378414	0.0000000	0.0000000	0.0000000	0.0000000
36	0.0000000	-2.378414	0.0000000	0.0000000	0.0000000
37	0.0000000	2.378414	0.0000000	0.0000000	0.0000000
38	0.0000000	0.0000000	-2.378414	0.0000000	0.0000000
39	0.0000000	0.0000000	2.378414	0.0000000	0.0000000
40	0.0000000	0.0000000	0.0000000	-2.378414	0.0000000
41	0.0000000	0.0000000	0.0000000	2.378414	0.0000000
42	0.0000000	0.0000000	0.0000000	0.0000000	-2.378414
43	0.0000000	0.0000000	0.0000000	0.0000000	2.378414

[3]: ccd.levels

	t_wall	CP3_y	T_0in	p_0in	T_e
Levels					
-2.378414	0.001000	-0.012558	400.000000	400000.000000	290.000000
-1.000000	0.003608	-0.008919	472.443974	573865.537712	321.875349
0.000000	0.005500	-0.006279	525.000000	700000.000000	345.000000
1.000000	0.007392	-0.003639	577.556026	826134.462288	368.124651
2.378414	0.010000	0.000000	650.000000	1000000.000000	400.000000

This DoE variables was then used as boundary conditions for the CFD(Computational Fluid Dynamics) numerical experiments. The 43 runs accounts for accounts for 429.6MB of data stored in a HDF5 file, from wich, the seleceted fields were choosen to build snapshot matrices

'Heat_Flux_UPPER_WALL', 'Mach', 'Pressure', 'Temperature', 'Temperature_Solid',
'Temperature_Solid_INNERWALL'

For each experiment a snapshot vector S_i , corresponding to the concatation of fluid and solid domain 2D numerical solutions were obtained.

$$S_i(T_{0in}, p_{0in}, t_{wall}, CP3_y) = \begin{bmatrix} q_{upperWall} \\ P_{fluid} \\ T_{fluid} \\ M_{fluid} \\ T_{solid} \\ T_{solidInner} \end{bmatrix}^{252840 \times 1}$$

, the snapshots S_i were then concatenated to form the snapshot matrix A of the Design of Experiment (DoE).

$$A = \begin{bmatrix} S_1^T \\ S_2^T \\ \vdots \\ S_N^T \end{bmatrix}^{43 \times 252840}$$

```
[4]: import sys
sys.path.append('/home/ppiper/Dropbox/local/github/frog')

from frog.datahandler import HDF5Handler

hfdh = HDF5Handler(
    datapath='/home/ppiper/Dropbox/local/github/explann/data/experimental_planning_T0in_
    ↪limit_clear/hf_transpose.h5',
    datasets = ['Heat_Flux_UPPER_WALL', 'Mach', 'Pressure', 'Temperature', 'Temperature_
    ↪Solid', 'Temperature_Solid_INNERWALL'],
)
```

2.2.14 Dependent variables (Order Reduction)

The high dimensional space of samples will subsequently be submitted to an order reduction, given by Principal Component Analysis (PCA), resulting into a more manageable, low-dimensional orthogonal basis. Subsequently, a factorial model will be fitted to predict the principal components as a function of the independent variables. The resultant model will then be applied to reconstruct flow field and heat transfer aspects for a designated test set of independent variables. This reconstructed output will be compared against the CFD solutions for error analysis.

The dependent variables are principal components of a POD(Proper Orthogonal Decomposition) of the snapshots matrix A

$$A = U \Sigma V^T$$

by truncating the diagonal matrix of singular values Σ , the snapshot matrix can be represented in terms of a reduced basis \tilde{U} and principal components \tilde{A}

$$\tilde{A} = \tilde{U} \tilde{\Lambda}$$

The data were then compressed using 10 principal components of the orthogonal projection, resulting in 24.3MB of data. An approximated 20-fold reduction in storage requirements.

```
[5]: from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA
from frog.normalization import Normalization, DataHandlerNormalization
import pickle

rom = Pipeline([
    ('scaler', Normalization(bounds=[-1,1])),
    ('pca', PCA(n_components=10)),
])

rom.fit(hfdh.data)
```

(continues on next page)

(continued from previous page)

```

results_pc = rom.transform(hfdh.data)

with open('/home/ppiper/Dropbox/local/github/explann/data/experimental_planning_T0in_
↳ limit_clear/rom.pkl','wb') as f:
    pickle.dump(rom,f)

with open('/home/ppiper/Dropbox/local/github/explann/data/experimental_planning_T0in_
↳ limit_clear/rom.pkl','rb') as f:
    rom = pickle.load(f)

```

The principal components are then joined to the CCD DoE to form full tables for factorial model construction.

```

[6]: import pandas as pd
results = {f'L{i}': results_pc[:,i] for i in range(results_pc.shape[1])}

ccd.append_results(results)
ccd.doe

```

```

[6]:
Index  t_wall  CP3_y  T_0in  p_0in  T_e  L0  L1  \
1      -1.000000 -1.000000 -1.000000 -1.000000 -1.000000  47.655208 -2.574525
2       1.000000 -1.000000 -1.000000 -1.000000 -1.000000  47.657308 -2.584284
3      -1.000000  1.000000 -1.000000 -1.000000 -1.000000  47.482853  2.523185
4       1.000000  1.000000 -1.000000 -1.000000 -1.000000  47.484867  2.512161
5      -1.000000 -1.000000  1.000000 -1.000000 -1.000000  47.629272 -2.501821
6       1.000000 -1.000000  1.000000 -1.000000 -1.000000  47.632879 -2.517772
7      -1.000000  1.000000  1.000000 -1.000000 -1.000000  47.456260  2.599557
8       1.000000  1.000000  1.000000 -1.000000 -1.000000  47.459997  2.581783
9      -1.000000 -1.000000 -1.000000  1.000000 -1.000000 -47.418552 -3.711689
10       1.000000 -1.000000 -1.000000  1.000000 -1.000000 -47.415844 -3.725914
11      -1.000000  1.000000 -1.000000  1.000000 -1.000000 -47.666183  3.624649
12       1.000000  1.000000 -1.000000  1.000000 -1.000000 -47.663805  3.609438
13      -1.000000 -1.000000  1.000000  1.000000 -1.000000 -47.451135 -3.618356
14       1.000000 -1.000000  1.000000  1.000000 -1.000000 -47.446400 -3.641930
15      -1.000000  1.000000  1.000000  1.000000 -1.000000 -47.698985  3.721687
16       1.000000  1.000000  1.000000  1.000000 -1.000000 -47.694650  3.696741
17      -1.000000 -1.000000 -1.000000 -1.000000  1.000000  47.655208 -2.574525
18       1.000000 -1.000000 -1.000000 -1.000000  1.000000  47.657308 -2.584284
19      -1.000000  1.000000 -1.000000 -1.000000  1.000000  47.482853  2.523185
20       1.000000  1.000000 -1.000000 -1.000000  1.000000  47.484867  2.512161
21      -1.000000 -1.000000  1.000000 -1.000000  1.000000  47.629272 -2.501821
22       1.000000 -1.000000  1.000000 -1.000000  1.000000  47.632879 -2.517772
23      -1.000000  1.000000  1.000000 -1.000000  1.000000  47.456260  2.599557
24       1.000000  1.000000  1.000000 -1.000000  1.000000  47.459997  2.581783
25      -1.000000 -1.000000 -1.000000  1.000000  1.000000 -47.418552 -3.711689
26       1.000000 -1.000000 -1.000000  1.000000  1.000000 -47.415844 -3.725914
27      -1.000000  1.000000 -1.000000  1.000000  1.000000 -47.666183  3.624649
28       1.000000  1.000000 -1.000000  1.000000  1.000000 -47.663805  3.609438
29      -1.000000 -1.000000  1.000000  1.000000  1.000000 -47.451135 -3.618356
30       1.000000 -1.000000  1.000000  1.000000  1.000000 -47.446400 -3.641930
31      -1.000000  1.000000  1.000000  1.000000  1.000000 -47.698985  3.721687
32       1.000000  1.000000  1.000000  1.000000  1.000000 -47.694650  3.696741
33       0.000000  0.000000  0.000000  0.000000  0.000000   0.025936 -0.038532

```

(continues on next page)

(continued from previous page)

34	-2.378414	0.000000	0.000000	0.000000	0.000000	0.022612	-0.015342
35	2.378414	0.000000	0.000000	0.000000	0.000000	0.029698	-0.056069
36	0.000000	-2.378414	0.000000	0.000000	0.000000	0.137596	-7.218266
37	0.000000	2.378414	0.000000	0.000000	0.000000	-0.381774	7.591583
38	0.000000	0.000000	-2.378414	0.000000	0.000000	0.059251	-0.135482
39	0.000000	0.000000	2.378414	0.000000	0.000000	-0.009178	0.056877
40	0.000000	0.000000	0.000000	-2.378414	0.000000	113.128140	-0.016841
41	0.000000	0.000000	0.000000	2.378414	0.000000	-113.070333	-0.076683
42	0.000000	0.000000	0.000000	0.000000	-2.378414	0.025936	-0.038532
43	0.000000	0.000000	0.000000	0.000000	2.378414	0.025936	-0.038532

	L2	L3	L4	L5	L6	L7	L8 \
Index							
1	1.054864	0.116466	0.006886	0.026542	-0.019255	0.012261	-0.003517
2	1.705080	-0.057877	0.004462	0.046149	-0.004435	-0.013983	0.008446
3	1.070312	0.115771	0.007093	0.020386	-0.025199	0.005370	-0.005530
4	1.748841	-0.031342	0.003471	0.026343	-0.011534	-0.007418	0.008019
5	-1.897897	0.122485	-0.010489	-0.061067	0.013869	0.004764	0.001125
6	-0.854666	-0.157822	-0.009886	-0.029965	0.015498	-0.008033	0.008307
7	-1.962213	0.058714	-0.004065	-0.022829	-0.001661	-0.012595	-0.013704
8	-0.875525	-0.179893	-0.005059	-0.016389	-0.003456	-0.008670	-0.003721
9	1.184190	0.148468	0.003752	-0.001800	0.000860	0.003282	-0.012650
10	2.135199	-0.085025	-0.001124	-0.016591	0.008043	-0.013796	-0.002655
11	1.288358	0.170554	-0.002338	-0.021060	0.008651	-0.001266	-0.003357
12	2.268540	-0.022280	-0.010885	-0.053166	0.020144	0.008198	0.006091
13	-2.496137	0.145204	0.006485	0.028797	0.001590	-0.001518	0.004297
14	-0.984372	-0.229059	0.005923	0.006421	-0.017370	0.012474	-0.000379
15	-2.469012	0.095408	0.006392	0.063492	0.008184	-0.021897	0.007073
16	-0.914726	-0.216737	0.001333	0.009757	-0.010209	0.027114	0.000185
17	1.054864	0.116466	0.006886	0.026542	-0.019255	0.012261	-0.003517
18	1.705080	-0.057877	0.004462	0.046149	-0.004435	-0.013983	0.008446
19	1.070312	0.115771	0.007093	0.020386	-0.025199	0.005370	-0.005530
20	1.748841	-0.031342	0.003471	0.026343	-0.011534	-0.007418	0.008019
21	-1.897897	0.122485	-0.010489	-0.061067	0.013869	0.004764	0.001125
22	-0.854666	-0.157822	-0.009886	-0.029965	0.015498	-0.008033	0.008307
23	-1.962213	0.058714	-0.004065	-0.022829	-0.001661	-0.012595	-0.013704
24	-0.875525	-0.179893	-0.005059	-0.016389	-0.003456	-0.008670	-0.003721
25	1.184190	0.148468	0.003752	-0.001800	0.000860	0.003282	-0.012650
26	2.135199	-0.085025	-0.001124	-0.016591	0.008043	-0.013796	-0.002655
27	1.288358	0.170554	-0.002338	-0.021060	0.008651	-0.001266	-0.003357
28	2.268540	-0.022280	-0.010885	-0.053166	0.020144	0.008198	0.006091
29	-2.496137	0.145204	0.006485	0.028797	0.001590	-0.001518	0.004297
30	-0.984372	-0.229059	0.005923	0.006421	-0.017370	0.012474	-0.000379
31	-2.469012	0.095408	0.006392	0.063492	0.008184	-0.021897	0.007073
32	-0.914726	-0.216737	0.001333	0.009757	-0.010209	0.027114	0.000185
33	-0.035311	-0.020620	-0.031631	0.018992	-0.033009	0.004036	-0.009210
34	-1.611487	0.417478	-0.036272	-0.035432	-0.028328	0.022534	0.024267
35	1.020713	-0.226146	-0.035953	-0.007553	-0.032945	0.004386	0.014950
36	-0.059227	-0.014131	0.189392	-0.011965	0.017133	0.001690	0.000903
37	0.028908	-0.014403	0.190904	-0.011537	0.015566	0.002630	-0.000132
38	3.737966	0.092381	-0.044229	-0.044475	-0.002733	-0.021055	0.002082
39	-3.682362	-0.139727	-0.048415	-0.052666	-0.000609	-0.014789	-0.004364

(continues on next page)

(continued from previous page)

40	0.340592	-0.011511	-0.054779	0.059510	0.105344	0.029156	-0.003415
41	0.329156	-0.028152	-0.069653	0.037104	0.058162	-0.005234	-0.002726
42	-0.035311	-0.020620	-0.031631	0.018992	-0.033009	0.004036	-0.009210
43	-0.035311	-0.020620	-0.031631	0.018992	-0.033009	0.004036	-0.009210

L9

Index

1	-0.003013
2	-0.001133
3	-0.004666
4	-0.007382
5	-0.008234
6	0.003685
7	0.005997
8	0.009605
9	0.009544
10	0.007812
11	0.001258
12	-0.006560
13	-0.008894
14	-0.001259
15	0.003627
16	-0.000820
17	-0.003013
18	-0.001133
19	-0.004666
20	-0.007382
21	-0.008234
22	0.003685
23	0.005997
24	0.009605
25	0.009544
26	0.007812
27	0.001258
28	-0.006560
29	-0.008894
30	-0.001259
31	0.003627
32	-0.000820
33	-0.002065
34	0.016765
35	0.010513
36	-0.000792
37	0.000395
38	-0.007991
39	-0.012773
40	0.004612
41	-0.003670
42	-0.002065
43	-0.002065

2.2.15 Fitting Factorial Model

The CCD factorial model accounts for second order terms in each L_i component of the Λ matrix.

```
[7]: from explann.models import FactorialModel
from explann.dataio import ImportString, ImportXLSX
from explann.plot import ParetoPlot
import matplotlib.pyplot as plt

expressions = {f'{key}': f'{key} ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_
↪ wall, 2) + np.power(T_0in, 2) + np.power(p_0in, 2) + np.power(CP3_y,2) + np.power(CP3_
↪ y, 2)' for key in results.keys()}

fm_ccd = FactorialModel(
    data = ccd.doe,
    functions = expressions,
    levels = ccd.levels,
)
expressions
```

```
[7]: {'L0': 'L0 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_
↪ 0in, 2) + np.power(p_0in, 2) + np.power(CP3_y,2) + np.power(CP3_y, 2)',
'L1': 'L1 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_
↪ 0in, 2) + np.power(p_0in, 2) + np.power(CP3_y,2) + np.power(CP3_y, 2)',
'L2': 'L2 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_
↪ 0in, 2) + np.power(p_0in, 2) + np.power(CP3_y,2) + np.power(CP3_y, 2)',
'L3': 'L3 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_
↪ 0in, 2) + np.power(p_0in, 2) + np.power(CP3_y,2) + np.power(CP3_y, 2)',
'L4': 'L4 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_
↪ 0in, 2) + np.power(p_0in, 2) + np.power(CP3_y,2) + np.power(CP3_y, 2)',
'L5': 'L5 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_
↪ 0in, 2) + np.power(p_0in, 2) + np.power(CP3_y,2) + np.power(CP3_y, 2)',
'L6': 'L6 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_
↪ 0in, 2) + np.power(p_0in, 2) + np.power(CP3_y,2) + np.power(CP3_y, 2)',
'L7': 'L7 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_
↪ 0in, 2) + np.power(p_0in, 2) + np.power(CP3_y,2) + np.power(CP3_y, 2)',
'L8': 'L8 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_
↪ 0in, 2) + np.power(p_0in, 2) + np.power(CP3_y,2) + np.power(CP3_y, 2)',
'L9': 'L9 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_
↪ 0in, 2) + np.power(p_0in, 2) + np.power(CP3_y,2) + np.power(CP3_y, 2)'}
```

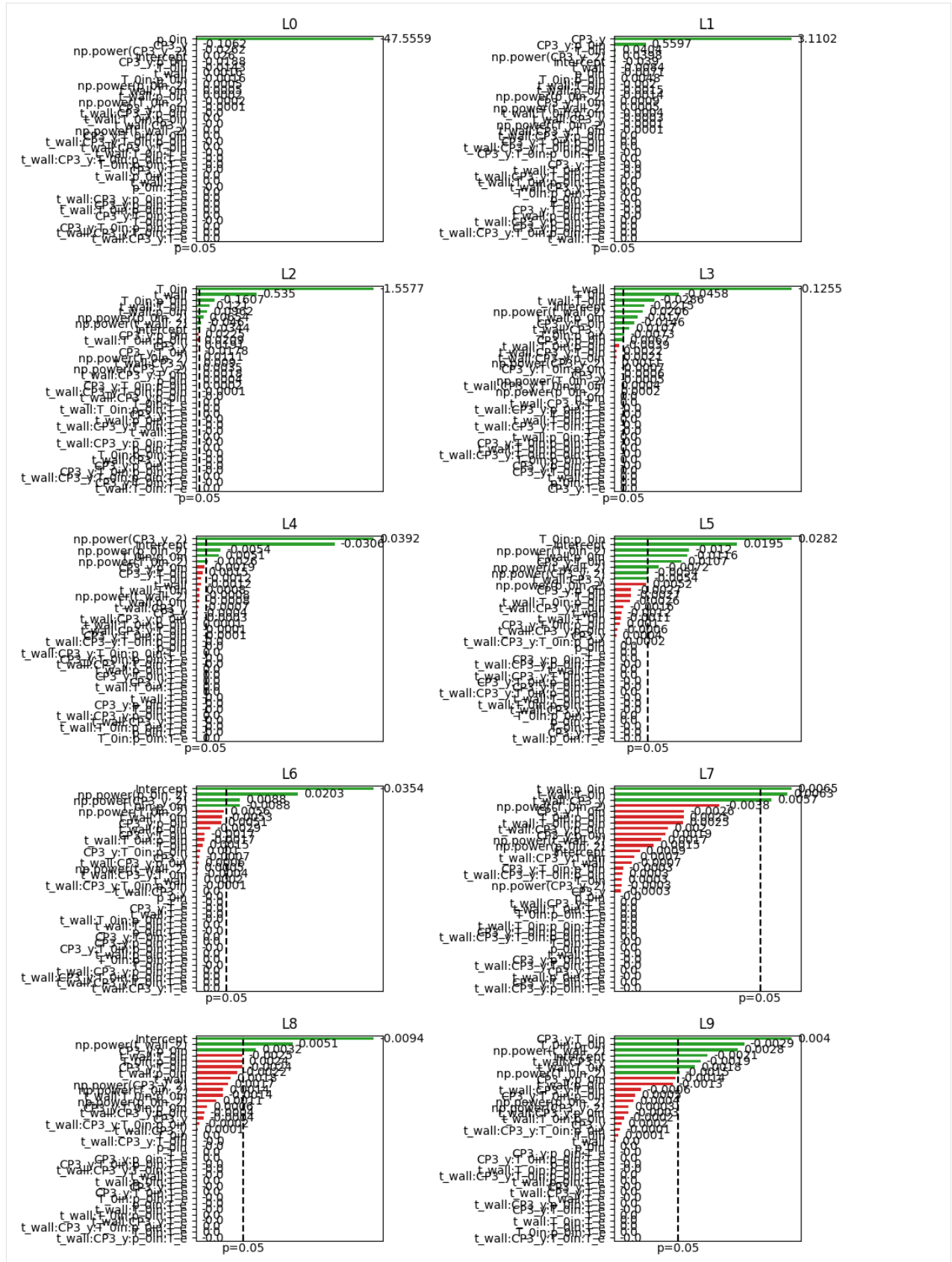
As can be viewed in pareto plots, not all terms are significant, so we can retaining fewer terms.

```
[8]: from explann.plot import ParetoPlot
import matplotlib.pyplot as plt

fig, ax = plt.subplots(5,2, figsize=(10,15))
ax = ax.flatten()

pp = ParetoPlot(fm_ccd)

pp.plot(ax=ax)
plt.tight_layout()
```



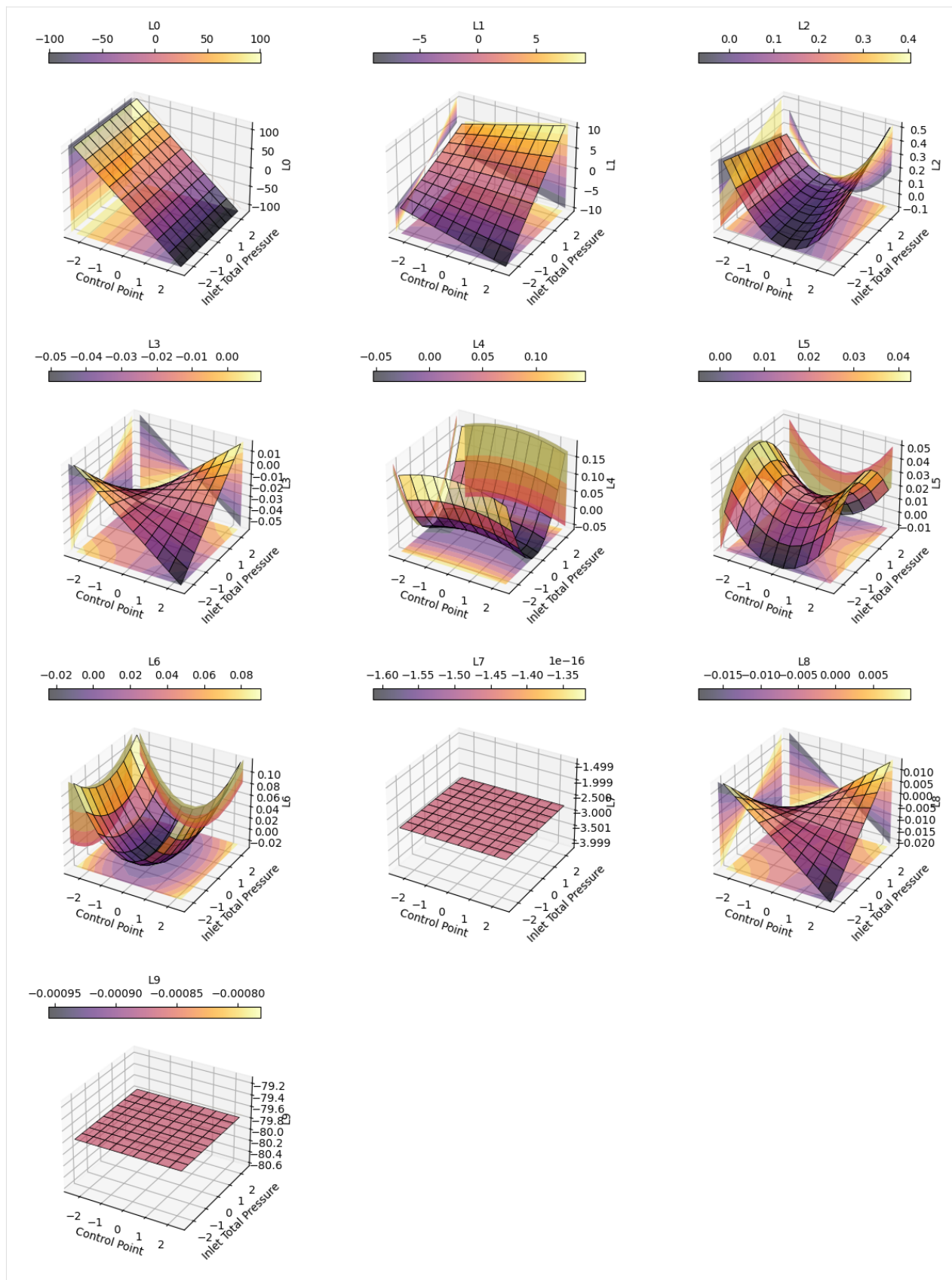
```
[9]: fm_ccd.get_significant_model_functions()
fm_ccd_full = fm_ccd
fm_ccd = fm_ccd.build_significant_models()
```

2.2.16 Resulting Model for principal components

```
[10]: from explann.plot import plot_surface

fig, ax = plt.subplots(4,3, figsize=(15,20),subplot_kw={"projection": "3d"})
ax = ax.flatten()

for i,axi in enumerate(ax):#fm_ccd.functions.items()):
    if i<len(fm_ccd.functions.items()):
        key = list(fm_ccd.functions.keys())[i]
        plot_surface(x='p_0in', y='CP3_y', z=key,
                      model=fm_ccd,
                      n_pts=10,
                      ax=ax[i],
                      other_params=dict(t_wall=0,T_0in=0,T_e=0),
                      labels=dict(xlabel='Control Point', ylabel='Inlet Total Pressure',
↪zlabel=key), cmap='inferno', scaled=False)
    else:
        ax[i].set_axis_off()
```



2.2.17 Test Model Accuracy

The model accuracy was tested by comparing flow reconstruction with full numerical solution for an unseeing set of independent variables

```
[11]: import numpy as np

def surrogate(model, variables, rom=None):
    independent_vars_coded = {}
    independent_vars = variables.copy()
    for var, value in independent_vars.items():
        independent_vars_coded[var] = np.interp(value, model.levels[f'{var}'].values,
↪model.levels.index.values)

    L_predicted = []
    for dependent in model.dependent_variables:
        L_predicted.append(model[dependent].predict(independent_vars_coded).item())

    L_predicted = np.array(L_predicted)

    if rom is not None:
        L_predicted = rom.inverse_transform(L_predicted)

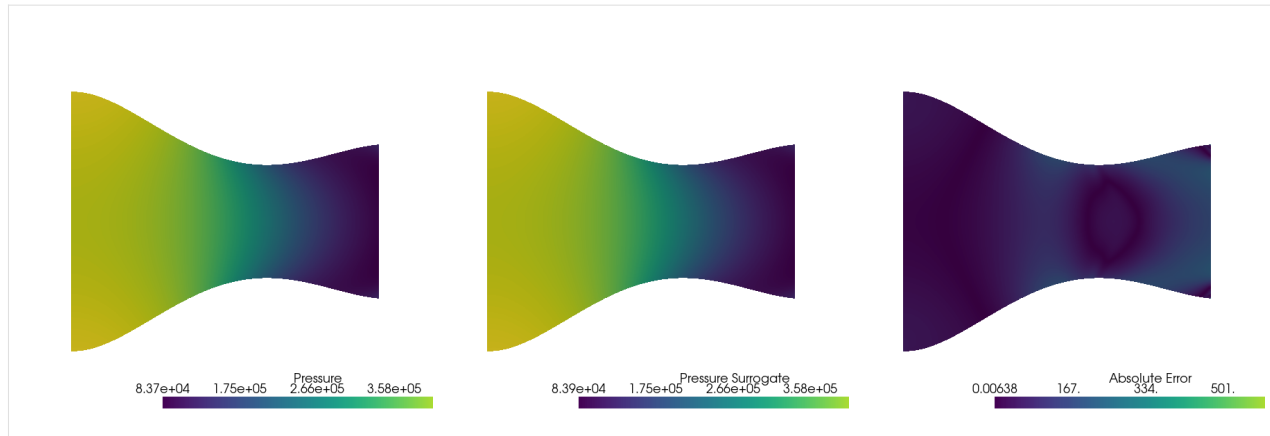
    return L_predicted

surrogate_solution = surrogate(
    model=fm_ccd_full,
    variables={
        't_wall': 0.006044329228014826,
        'CP3_y': -0.001277584765410799,
        'T_0in': 616.0,
        'p_0in': 456429.13834591914,
        'T_e': 318.0
    },
    rom=rom)

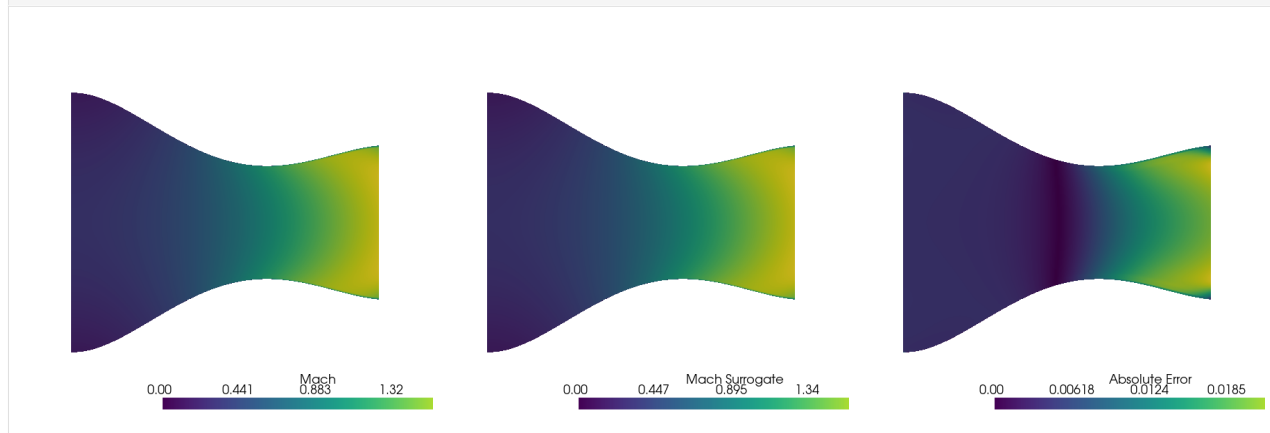
inverse_transform
```

```
[12]: from plot_2d import plot_property
```

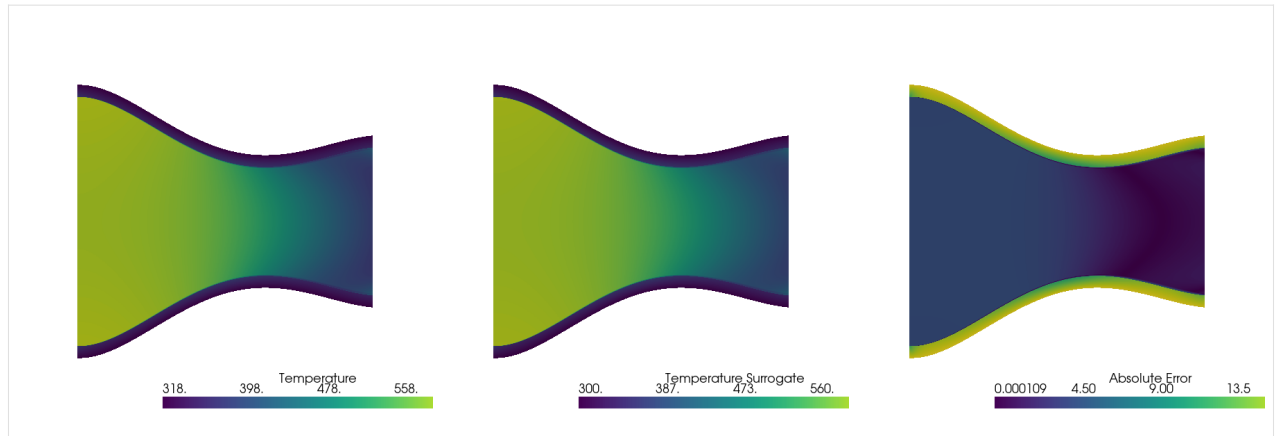
```
[13]: plot_property(
    prop='Pressure',
    unseeing_path='/home/ppiper/Dropbox/local/github/explann/data/lhs/10/SU2/outputs/cht_
↪setupSU2.vtm',
    surrogate_solution=surrogate_solution,
    hfdh=hfdh,
    solid=False)
```



```
[14]: plot_property(
    prop='Mach',
    unseeing_path='/home/ppiper/Dropbox/local/github/explann/data/lhs/10/SU2/outputs/cht_
    ↪ setupSU2.vtm',
    surrogate_solution=surrogate_solution,
    hfdh=hfdh,
    solid=False)
```



```
[15]: plot_property(
    prop='Temperature',
    unseeing_path='/home/ppiper/Dropbox/local/github/explann/data/lhs/10/SU2/outputs/cht_
    ↪ setupSU2.vtm',
    surrogate_solution=surrogate_solution,
    hfdh=hfdh,
    solid=True)
```

2.2.18 References

Back, L.H., Massier, P.F. and Gier, H.L., 1964. “Convective heat transfer in a convergent-divergent nozzle”. *International Journal of Heat and Mass Transfer*, Vol. 7, pp. 549–568.

Economon, T.D., Palacios, F., Copeland, S.R., Lukaczyk, T.W. and Alonso, J.J., 2016. “SU2: An open-source suite for multiphysics simulation and design”. *AIAA Journal*, Vol. 54, No. 3, pp. 828–846. doi:10.2514/1.j053813.

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

e

`explann`, [7](#)
`explann.dataio`, [5](#)
`explann.models`, [6](#)

INDEX

A

`add_categorical()` (in module *explann.models*), 7
`anova()` (*explann.models.FactorialModel* method), 6

B

`BaseImport` (class in *explann.dataio*), 5
`BaseModel` (class in *explann.models*), 6
`build_significant_models()` (ex-
 plann.models.FactorialModel method), 6

D

`data` (*explann.dataio.BaseImport* attribute), 5
`decode_variables()` (*explann.models.FactorialModel*
 method), 6
`delimiter` (*explann.dataio.BaseImport* attribute), 5
`dependent_variables` (ex-
 plann.models.FactorialModel property),
 6

E

`encode_variables()` (*explann.models.FactorialModel*
 method), 6
`explann`
 module, 7
`explann.dataio`
 module, 5
`explann.models`
 module, 6

F

`FactorialModel` (class in *explann.models*), 6
`fit()` (*explann.models.FactorialModel* method), 6
`function_names` (*explann.models.FactorialModel* prop-
 erty), 6

G

`get_categorical_functions()` (ex-
 plann.models.FactorialModel method), 6
`get_categorical_model()` (ex-
 plann.models.FactorialModel method), 6
`get_significant_model_functions()` (ex-
 plann.models.FactorialModel method), 6

`get_significant_terms()` (ex-
 plann.models.FactorialModel method), 6

I

`ImportString` (class in *explann.dataio*), 6
`ImportXLSX` (class in *explann.dataio*), 6
`independent_variables` (ex-
 plann.models.FactorialModel property),
 7

L

`lack_of_fit()` (*explann.models.FactorialModel*
 method), 7
`levels` (*explann.dataio.BaseImport* attribute), 5

M

module
 explann, 7
 explann.dataio, 5
 explann.models, 6

P

`parse_levels()` (*explann.dataio.BaseImport* method),
 5, 6
`parse_levels_from_string()` (ex-
 plann.dataio.BaseImport method), 5, 6
`parse_levels_from_xlsx()` (ex-
 plann.dataio.BaseImport method), 6
`parsed_data` (*explann.dataio.BaseImport* attribute), 5
`predict()` (*explann.models.FactorialModel* method), 7
`predict_rescaled()` (*explann.models.FactorialModel*
 method), 7
`print_equation()` (*explann.models.FactorialModel*
 method), 7

S

`summary()` (*explann.models.FactorialModel* method), 7

W

`working_lack_of_fit()` (ex-
 plann.models.FactorialModel method), 7