Optimize the power conversion efficiency of organic photovoltaic solar cells

Introduction

Organic photovoltaic (OPV) are 'plastic' solar cells that can be made cheaply and easily as you can use techniques like roll to roll printing, inject printing and spray coating. Current generation solar cells take several years of use before they payback the energy required in their manufacture, OPVs are so efficient that their energy payback is only 24hours. Power conversion efficiencies (PCEs) of OPVs are now around 14%. To commercialise them, we need to figure out how best to manufacture them.

Organic photovoltaic devices have a sandwich architecture. The bottom layers Al/Mg and LiF are the bottom electrode. The important part is the bulk hetereojunction, shown in red in the figure below, which comprises of a low band gap polymer which is the electron donor and fullerene which is the electron acceptor. Addition of an additive helps with forming and bridging separate nanodomains of donor and accceptor. Solar cells work by using light to form an exciton which then separates into an electron-hole pair and you want these to be separated from each other, which is why you want separate nanodomain of donor and acceptor. The top of the solar cell is PEDOT:PSS (a conducting polymer) and ITO (indium tin oxide), a see-through electrode, which together act as the top electrode.

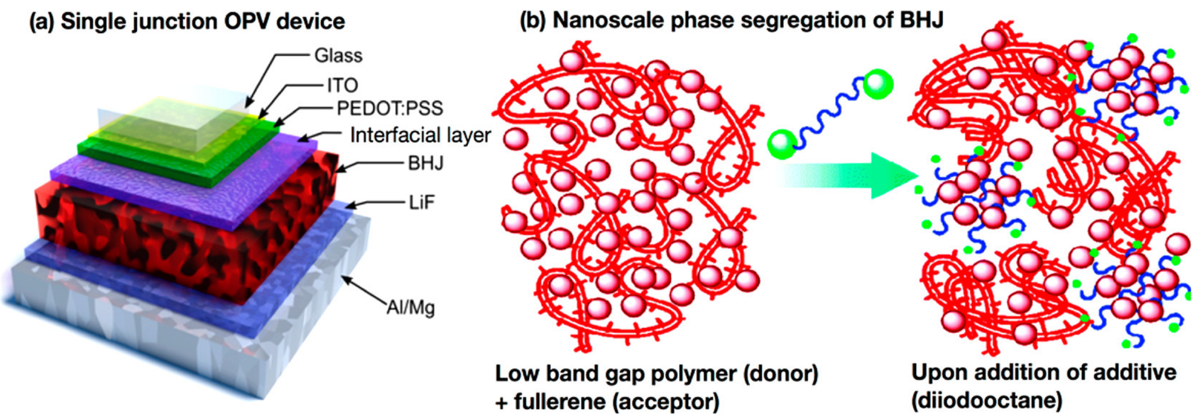


Figure 1. (a) Schematic of single junction organic photovoltaic (OPV) devices, showing the bulk heterojunction (BHJ; in red), and the multiple interfacial layers in the device. (b) Schematic of BHJ morphology: in this case, a low band gap polymer donor and a fullerene acceptor undergoing nanoscale phase segregation into discrete nanoscale domains of donor and acceptor. The use of an additive is often purported to assist in nanodomain formation, as shown here. Taken from [ACS Nano 2018, 12, 7434−7444]

The task

The task is to optimise the construction of this type of solar cell. Donor weight percentage is a measure of the ratio of donor to acceptor in the heterojunction. Total solution concentration is the concentration of the spin-coating solution. Bulk heterojunction spin-case speed is a measure of how fast you spin the device when coating it with the bulk heterojunction mixture. Processing additive is the amount of additive (diiodooctane) added to the mixture. The thickness of a spun film is determined by the spin speed, solvent vapour pressure and solution viscosity, as both the donor weight percentage and total solution concentration can affect viscosity, the first three factors can all affect the thickness of the final BHJ layer. The additive (diiodooctane) increases the drying time for the film, helping to separate the hetereojunction out into nanodomains of donor and acceptor rich areas.

Factors selected:

Name	Factors	Factor range	No. of levels
Donor	Donor weight percentage	10-55 (wt %)	4
Conc.	Total solution concentration	10-25 (mg/mL)	4
Spin	Bulk heterojunction spin-case speed	600 - 3000 (rpm)	4
Add.	Processing additive	0-12 (vol %)	4

We shall use the shortened names from the table above.

Files:

1. `solar_cells_1.csv` results from the first experiment, fractional factorial, 4 factors and 4 levels, here we have 16 experiments, one failed to solidify.
2. `'solar_cells_2.csv'` has results from the second experiment, a fractional factorial, 3 factors and 3 levels. This covers a smaller range.

The data is taken from: "How To Optimize Materials and Devices via Design of Experiments and Machine Learning: Demonstration Using Organic Photovoltaics", Bing Cao, Lawrence A. Adutwum, Anton O. Oliynyk, Erik J. Luber, Brian C. Olsen, Arthur Mar, and Jillian M. Buriak, ACS Nano 2018, 12, 7434−7444

First we import our packages

In [1]:
```
# for dataframes
import pandas as pd

# for pictures
import matplotlib.pyplot as plt
# for maths
import numpy as np

## Some code in doenut needs updating, so use this to ignore the warnings
import warnings
warnings.filterwarnings('ignore')

# make sure these are in the same directory as this file
import doenut
import designer
```

Read in the first experiment's data

In [2]:
```
df=pd.read_csv('solar_cells_1.csv')
df
```

Out[2]:

	experiment #	donor percentage	total concentration	spin speed	additive	PCE	std of PCE (%)	number of devices
0	NaN	% (wt)	mg/mL	rpm	vol %	%	NaN	NaN
1	1-1	10	20	3000	2	0.05	5.0	14.0
2	1-2	10	25	1000	8	3.24	11.0	10.0
3	1-3	10	10	600	0	0.016	16.0	14.0
4	1-4	10	15	2000	12	0.0004	4.0	10.0
5	1-5	25	20	600	12	7.14	13.0	8.0
6	1-6	25	15	1000	2	3.22	32.0	8.0
7	1-7	25	10	3000	8	0.00033	7.0	14.0
8	1-8	25	25	2000	0	7.21	17.0	11.0
9	1-9	40	10	1000	12	1.85	5.0	3.0
10	1-10	40	20	2000	8	6.16	28.0	12.0
11	1-11	40	25	600	2	3.9	8.0	11.0
12	1-12	40	15	3000	0	2.27	35.0	9.0
13	1-13	55	10	2000	2	1.16	4.0	3.0
14	1-14	55	15	600	8	3.18	12.0	10.0
15	1-15	55	20	1000	0	3.89	10.0	13.0
16	1-16	55	25	3000	12	NaN	NaN	NaN

Set up input and responsese dataframes

We must drop the last experiment, as these devices didn't set.

In [3]:

```
inputs = pd.DataFrame({
    'Donor %': [float(x) for x in df.iloc[1:-1,1]],
    'Conc.': [float(x) for x in df.iloc[1:-1,2]],
    'Spin': [float(x) for x in df.iloc[1:-1,3]],
    'Add.': [float(x) for x in df.iloc[1:-1,4]]})
inputs
```

Out[3]:

	Donor %	Conc.	Spin	Add.
0	10.0	20.0	3000.0	2.0
1	10.0	25.0	1000.0	8.0
2	10.0	10.0	600.0	0.0
3	10.0	15.0	2000.0	12.0
4	25.0	20.0	600.0	12.0
5	25.0	15.0	1000.0	2.0
6	25.0	10.0	3000.0	8.0
7	25.0	25.0	2000.0	0.0
8	40.0	10.0	1000.0	12.0

	Donor %	Conc.	Spin	Add.
9	40.0	20.0	2000.0	8.0
10	40.0	25.0	600.0	2.0
11	40.0	15.0	3000.0	0.0
12	55.0	10.0	2000.0	2.0
13	55.0	15.0	600.0	8.0
14	55.0	20.0	1000.0	0.0

In [4]:
```
responses = pd.DataFrame({'PCE':  [float(x) for x in df['PCE'][1:-1]]})
responses
```

Out[4]:

	PCE
0	0.05000
1	3.24000
2	0.01600
3	0.00040
4	7.14000
5	3.22000
6	0.00033
7	7.21000
8	1.85000
9	6.16000
10	3.90000
11	2.27000
12	1.16000
13	3.18000
14	3.89000

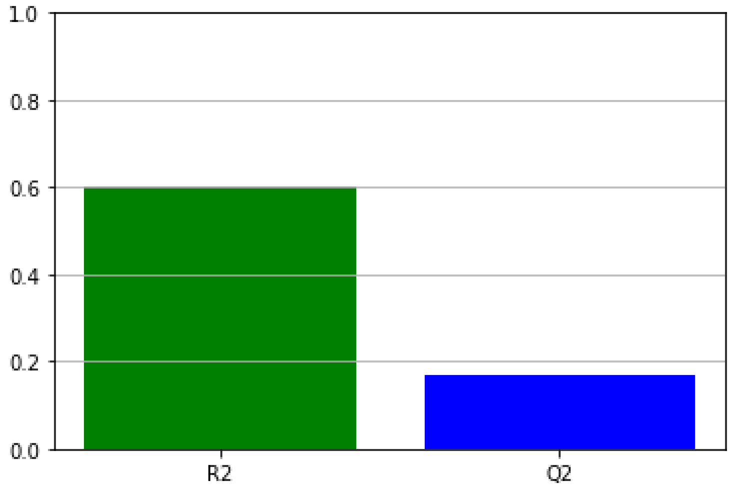
Task 1. Create a linear (main factors only) model

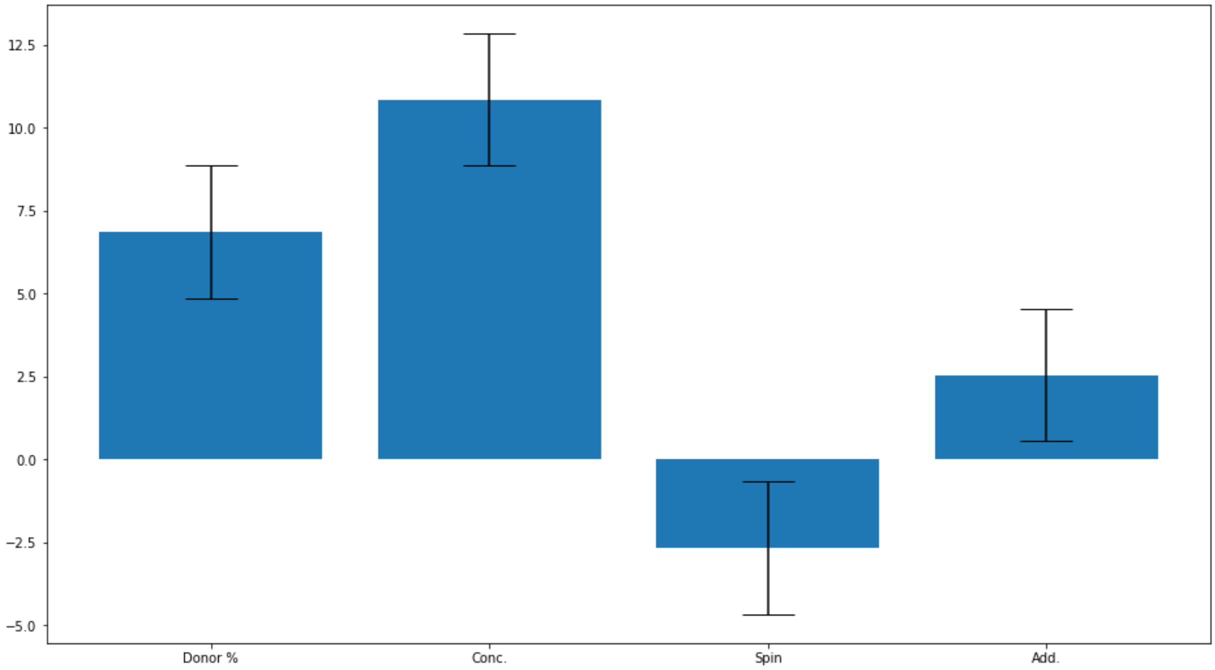
Create a linear model, i.e. a model that has just the main effects (also known as a first order model or main effects model) Fit your linear model to the first experiment's data and calculate R2 and Q2 for your fitted model. Then answer the questions.

In [5]:
```
# this selects which columns in inputs to use
input_selector = range(len(inputs.columns))

this_model, R2, temp_tuple, _ = doenut.calulate_R2_and_Q2_for_models(
                                inputs,
                                # input dataframe
                                responses,
```

```
                                  # responses dataframe
                                  input_selector=input_selector,
                                  # which columns in input to fit
                                  response_selector=[0],
                                  # This selects the zeroth column of responses
                                  use_scaled_inputs=True,
                                  # we scale the inputs so the coefficients
                                  # are comparable
                                  do_scaling_here=True
                                  # scale inputs inside this function
                                  )


new_model, predictions, ground_truth, coeffs, R2s, R2, Q2= temp_tuple
```

```
Input terms are ['Donor %', 'Conc.', 'Spin', 'Add.']
Input Responses are ['PCE']

Selected Response is PCE
Selected input terms:   ['Donor %', 'Conc.', 'Spin', 'Add.']
Averaging replicates
Input data is 15 points long
We are using 15 data points
Left out data point 0:  R2 = 0.627      Ave. Error = -2.86
Left out data point 1:  R2 = 0.643      Ave. Error = -2.3
Left out data point 2:  R2 = 0.568      Ave. Error = 1.59
Left out data point 3:  R2 = 0.604      Ave. Error = -2.3
Left out data point 4:  R2 = 0.638      Ave. Error = 3.78
Left out data point 5:  R2 = 0.627      Ave. Error = 1.6
Left out data point 6:  R2 = 0.558      Ave. Error = -0.0576
Left out data point 7:  R2 = 0.643      Ave. Error = 3.87
Left out data point 8:  R2 = 0.599      Ave. Error = -0.189
Left out data point 9:  R2 = 0.599      Ave. Error = 2.3
Left out data point 10: R2 = 0.672      Ave. Error = -3.01
Left out data point 11: R2 = 0.607      Ave. Error = 0.816
Left out data point 12: R2 = 0.591      Ave. Error = -0.533
Left out data point 13: R2 = 0.619      Ave. Error = -1.38
Left out data point 14: R2 = 0.615      Ave. Error = -1.43
R2 overall is 0.604
Mean of test set: 2.885782
Mean being used: 2.885782
Sum of squares of the residuals (explained variance) is 72.3948507524448
Sum of squares total (total variance) is 87.23748999604001
Q2 is 0.17
Response PCE R2 is 0.604
Input selector was range(0, 4)
Input_selector was: [0, 1, 2, 3]
Average coefficients are: [ 6.85475012 10.85602224 -2.66214206  2.5458804 ]
Coefficient labels are: ['Donor %', 'Conc.', 'Spin', 'Add.']
```





Task 2. Create a quadratic, parsimonious and hierarchical model

Task 2. Create a **quadratic**, **parsimonious** and **hierarchical** model. Starting with a quadratic model, and making sure that all models are hierarchical, optimise the model by removing **only** the **statistically insignificant** terms. Keep a note of the terms removed and teh R^2 and Q^2 values.

First we must expand the input dataframe to include the higher order terms.

In [6]:
```
sat_source_list = []
source_list = []
sat_inputs_orig, sat_source_list = doenut.add_higher_order_terms(
    inputs,
    add_squares=True,
    add_interactions=True,
    column_list=[])
```

```
Input array has columns ['Donor %', 'Conc.', 'Spin', 'Add.']
Adding square terms:
Donor %**2
Conc.**2
Spin**2
Add.**2
Adding interaction terms:
Donor %*Conc.
Donor %*Spin
Donor %*Add.
Conc.*Spin
Conc.*Add.
Spin*Add.
```

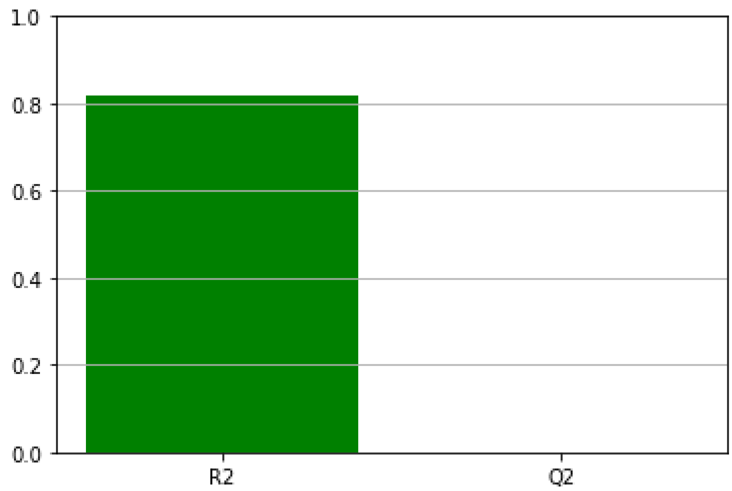
Full saturated quadratic model:

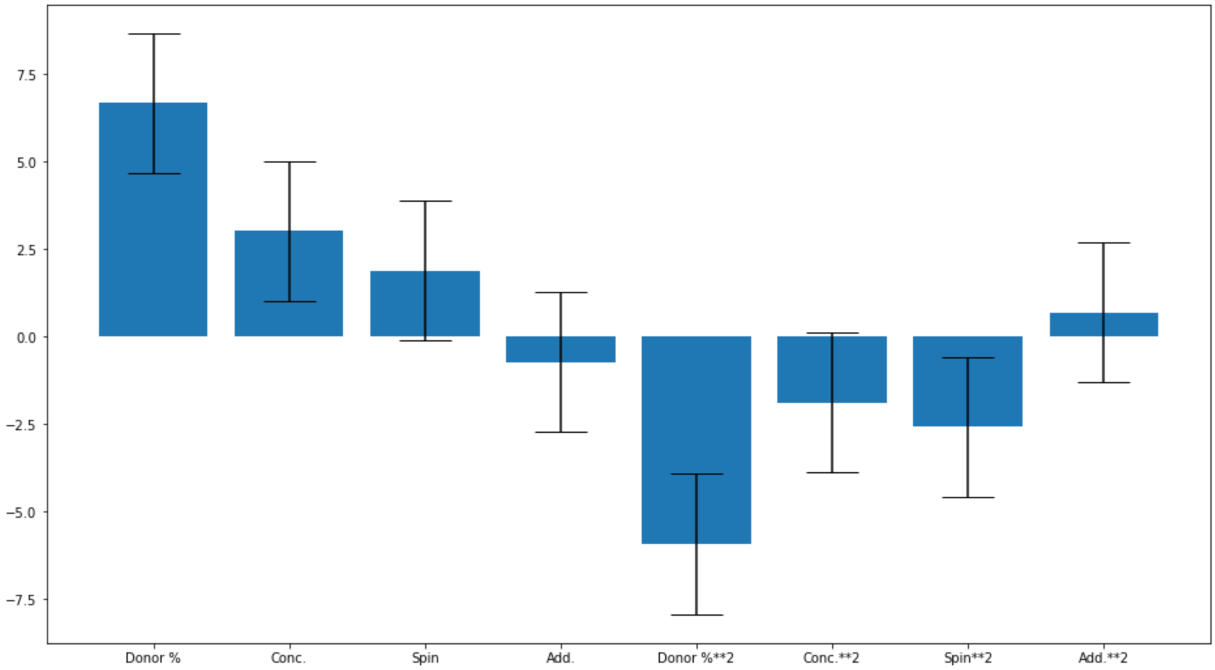
This contains all the main terms and all the square terms.

In [7]:
```
input_selector = [0, 1, 2, 3,
                  4, 5, 6, 7]
```

```
scaled_model, R2, temp_tuple, _ =doenut.tune_model(
                sat_inputs_orig,
                responses,
                input_selector=input_selector,
                response_selector=[0]
              )

new_model, predictions, ground_truth, coeffs, R2s, R2, Q2= temp_tuple
```

```
Input terms are ['Donor %', 'Conc.', 'Spin', 'Add.', 'Donor %**2', 'Conc.**2', 'Spin
**2', 'Add.**2', 'Donor %*Conc.', 'Donor %*Spin', 'Donor %*Add.', 'Conc.*Spin', 'Con
c.*Add.', 'Spin*Add.']
Input Responses are ['PCE']

Selected Response is PCE
Selected input terms:   ['Donor %', 'Conc.', 'Spin', 'Add.', 'Donor %**2', 'Conc.**
2', 'Spin**2', 'Add.**2']
Have found no replicates
Input data is 15 points long
We are using 15 data points
Left out data point 0:  R2 = 0.799      Ave. Error = -0.849
Left out data point 1:  R2 = 0.816      Ave. Error = 0.19
Left out data point 2:  R2 = 0.831      Ave. Error = 3.42
Left out data point 3:  R2 = 0.869      Ave. Error = -4.02
Left out data point 4:  R2 = 0.872      Ave. Error = 4.05
Left out data point 5:  R2 = 0.846      Ave. Error = -3.35
Left out data point 6:  R2 = 0.796      Ave. Error = 0.587
Left out data point 7:  R2 = 0.838      Ave. Error = 3.59
Left out data point 8:  R2 = 0.842      Ave. Error = -2.39
Left out data point 9:  R2 = 0.8        Ave. Error = 1.37
Left out data point 10: R2 = 0.92       Ave. Error = -4.38
Left out data point 11: R2 = 0.816      Ave. Error = -0.473
Left out data point 12: R2 = 0.819      Ave. Error = 1.49
Left out data point 13: R2 = 0.83       Ave. Error = 1.78
Left out data point 14: R2 = 0.817      Ave. Error = -0.841
R2 overall is 0.815
Mean of test set: 2.885782
Mean being used: 2.885782
Sum of squares of the residuals (explained variance) is 102.55309297319681
Sum of squares total (total variance) is 87.23748999604001
Q2 is -0.176
Response PCE R2 is 0.816
Input selector was [0, 1, 2, 3, 4, 5, 6, 7]
Input_selector was: [0, 1, 2, 3, 4, 5, 6, 7]
Average coefficients are: [ 6.68270726  3.02883412  1.87771844 -0.72988262 -5.932714
32 -1.89506711
 -2.58602063  0.69844306]
Coefficient labels are: ['Donor %', 'Conc.', 'Spin', 'Add.', 'Donor %**2', 'Conc.**
2', 'Spin**2', 'Add.**2']
```



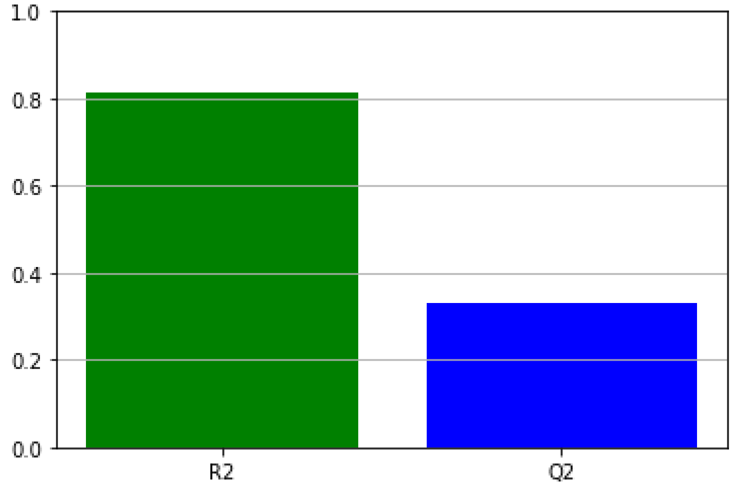


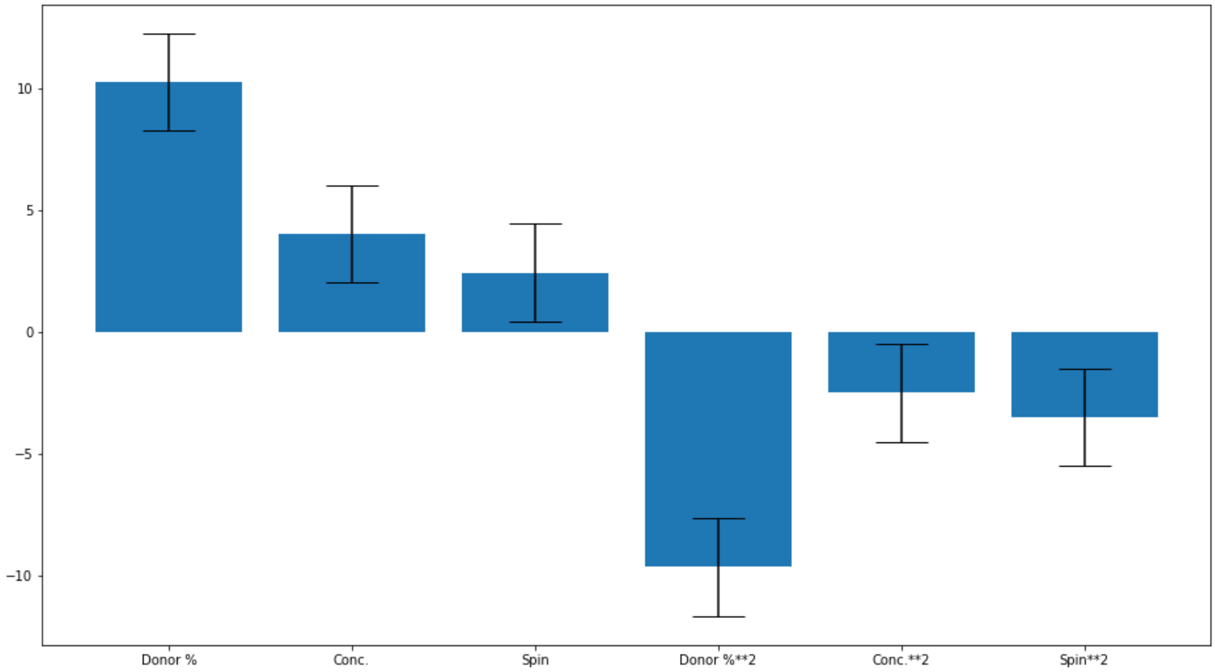
In [8]:

```
input_selector = [0, 1, 2,
                  4, 5, 6]
scaled_model, R2, temp_tuple, _ =doenut.tune_model(
                sat_inputs_orig,
                responses,
                input_selector=input_selector,
                response_selector=[0]
              )

new_model, predictions, ground_truth, coeffs, R2s, R2, Q2= temp_tuple
```

```
Input terms are ['Donor %', 'Conc.', 'Spin', 'Add.', 'Donor %**2', 'Conc.**2', 'Spin
**2', 'Add.**2', 'Donor %*Conc.', 'Donor %*Spin', 'Donor %*Add.', 'Conc.*Spin', 'Con
c.*Add.', 'Spin*Add.']
Input Responses are ['PCE']

Selected Response is PCE
Selected input terms:   ['Donor %', 'Conc.', 'Spin', 'Donor %**2', 'Conc.**2', 'Spin
**2']
Have found no replicates
Input data is 15 points long
We are using 15 data points
Left out data point 0:  R2 = 0.797      Ave. Error = -0.906
Left out data point 1:  R2 = 0.812      Ave. Error = -0.04
Left out data point 2:  R2 = 0.812      Ave. Error = 1.92
Left out data point 3:  R2 = 0.833      Ave. Error = -2.57
Left out data point 4:  R2 = 0.855      Ave. Error = 3.23
Left out data point 5:  R2 = 0.833      Ave. Error = -1.71
Left out data point 6:  R2 = 0.792      Ave. Error = 0.229
Left out data point 7:  R2 = 0.824      Ave. Error = 2.92
Left out data point 8:  R2 = 0.824      Ave. Error = -1.39
Left out data point 9:  R2 = 0.789      Ave. Error = 0.757
Left out data point 10: R2 = 0.919      Ave. Error = -4.4
Left out data point 11: R2 = 0.812      Ave. Error = -0.16
Left out data point 12: R2 = 0.814      Ave. Error = 1.33
Left out data point 13: R2 = 0.821      Ave. Error = 1.17
Left out data point 14: R2 = 0.812      Ave. Error = -0.467
R2 overall is 0.813
Mean of test set: 2.885782
Mean being used: 2.885782
Sum of squares of the residuals (explained variance) is 58.245326009154304
Sum of squares total (total variance) is 87.23748999604001
Q2 is 0.332
Response PCE R2 is 0.813
Input selector was [0, 1, 2, 4, 5, 6]
```

```
Input_selector was: [0, 1, 2, 3, 4, 5]
Average coefficients are: [10.24376537  4.00601457  2.41963174 -9.66908521 -2.524063
76 -3.51264286]
Coefficient labels are: ['Donor %', 'Conc.', 'Spin', 'Donor %**2', 'Conc.**2', 'Spin
**2']
```





15 datapoints so 14 DoF.

Starting from quadratic model

No. of terms	DoF	term removed	factor removed	R^2	Q^2
8	6			0.815	-0.176
7	7	8	'Add.**2	0.813	0.0863
6	8	3	Add.	0.813	0.332

this is the model with no statistically insignificant terms. It's heirarchical.

The Q2 is better than the main effects only model

Task 3. Create a parsimonious interaction model

Create hierarchical parsimonious interaction model. Starting with a interaction model, and making sure that all models are hierarchical, optimise the model by removing only the statistically insignificant terms. Keep a note of the terms removed and the Q^2 and R^2 values.

In [ ]:
```
input_selector = [0, 1, 2, 3,
                  9, 10]


scaled_model, R2, temp_tuple, _ =doenut.tune_model(
                sat_inputs_orig,
                responses,
                input_selector=input_selector,
                response_selector=[0]
              )

new_model, predictions, ground_truth, coeffs, R2s, R2, Q2= temp_tuple
```

15 terms so 14 DoF

Starting from square model

No. of terms	DoF	term removed	factor removed	R^2	Q^2
10	4			0.811	-1.79
9	5	12	`Conc.*Add.`	0.813	0.0863
8	6	13	`'Spin*Add.'`	0.798	-0.555
7	7	11	`Conc*Spin.`	0.777	0.0133
6	8	8	`Donor*Conc.`	0.761	0.315

this is the model with no statistically insignificant terms. It's heirarchical.

The Q2 is better than the main effects only model, but not as good as the square terms.

Task 4: Combine data from both experiments and train a parsimonious model

In [9]:
```
df=pd.read_csv('solar_cells_2.csv')
df
```

Out[9]:

	experiment #	donor %	total concentration	spin speed	PCE	thickness	number of devices
0	NaN	wt %	mg/mL	rpm	%	nm	NaN
1	2-1	20	20	1500	6.32	73	5.0
2	2-2	27	20	1500	7.21	77	11.0
3	2-3	20	25	1500	6.83	126	6.0
4	2-4	27	25	1500	6.96	131	6.0
5	2-5	20	23	1000	7.77	109	4.0
6	2-6	27	23	1000	6.87	136	4.0
7	2-7	20	23	2000	6.43	76	8.0
8	2-8	27	23	2000	7.65	88	7.0

	experiment #	donor %	total concentration	spin speed	PCE	thickness	number of devices
9	2-9	25	20	1000	7.43	115	4.0
10	2-10	25	25	1000	6.88	135	8.0
11	2-11	25	20	2000	7.32	104	7.0
12	2-12	25	25	2000	7.21	126	8.0
13	2-13	25	23	1500	7.4	129	7.0

In [10]:

```
inputs_2 = pd.DataFrame({
    'Donor %': [float(x) for x in df.iloc[1:-1,1]],
    'Conc.': [float(x) for x in df.iloc[1:-1,2]],
    'Spin': [float(x) for x in df.iloc[1:-1,3]]})
inputs_2
```

Out[10]:

	Donor %	Conc.	Spin
0	20.0	20.0	1500.0
1	27.0	20.0	1500.0
2	20.0	25.0	1500.0
3	27.0	25.0	1500.0
4	20.0	23.0	1000.0
5	27.0	23.0	1000.0
6	20.0	23.0	2000.0
7	27.0	23.0	2000.0
8	25.0	20.0	1000.0
9	25.0	25.0	1000.0
10	25.0	20.0	2000.0
11	25.0	25.0	2000.0

In [11]:

```
responses_2 = pd.DataFrame({'PCE':  [float(x) for x in df['PCE'][1:-1]]})
responses_2
```

Out[11]:

	PCE
0	6.32
1	7.21
2	6.83
3	6.96
4	7.77
5	6.87
6	6.43
7	7.65
8	7.43
9	6.88

	PCE
10	7.32
11	7.21

In [12]:
```
inputs[['Donor %', 'Conc.', 'Spin']]
```

Out[12]:

	Donor %	Conc.	Spin
0	10.0	20.0	3000.0
1	10.0	25.0	1000.0
2	10.0	10.0	600.0
3	10.0	15.0	2000.0
4	25.0	20.0	600.0
5	25.0	15.0	1000.0
6	25.0	10.0	3000.0
7	25.0	25.0	2000.0
8	40.0	10.0	1000.0
9	40.0	20.0	2000.0
10	40.0	25.0	600.0
11	40.0	15.0	3000.0
12	55.0	10.0	2000.0
13	55.0	15.0	600.0
14	55.0	20.0	1000.0

In [13]:
```
new_inputs = pd.concat([inputs[['Donor %', 'Conc.', 'Spin']], inputs_2], axis=0)
new_responses = pd.concat([responses, responses_2], axis=0)
```

In [14]:
```
new_inputs
```

Out[14]:

	Donor %	Conc.	Spin
0	10.0	20.0	3000.0
1	10.0	25.0	1000.0
2	10.0	10.0	600.0
3	10.0	15.0	2000.0
4	25.0	20.0	600.0
5	25.0	15.0	1000.0
6	25.0	10.0	3000.0
7	25.0	25.0	2000.0
8	40.0	10.0	1000.0
9	40.0	20.0	2000.0
10	40.0	25.0	600.0

	Donor %	Conc.	Spin
11	40.0	15.0	3000.0
12	55.0	10.0	2000.0
13	55.0	15.0	600.0
14	55.0	20.0	1000.0
0	20.0	20.0	1500.0
1	27.0	20.0	1500.0
2	20.0	25.0	1500.0
3	27.0	25.0	1500.0
4	20.0	23.0	1000.0
5	27.0	23.0	1000.0
6	20.0	23.0	2000.0
7	27.0	23.0	2000.0
8	25.0	20.0	1000.0
9	25.0	25.0	1000.0
10	25.0	20.0	2000.0
11	25.0	25.0	2000.0

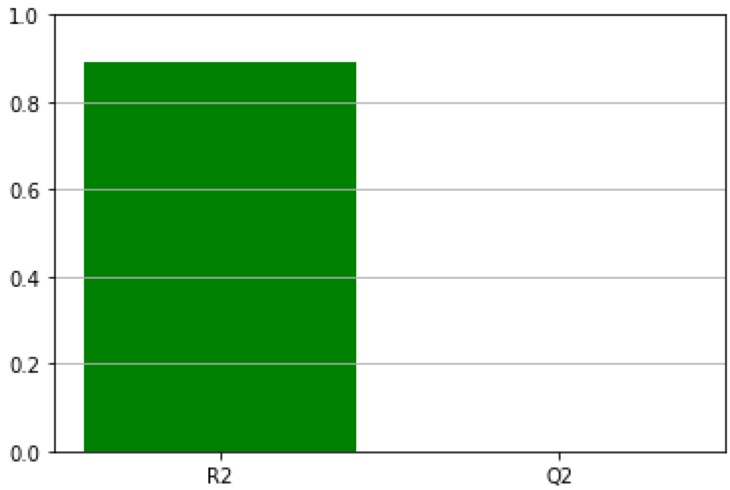
In [15]:
```
sat_source_list = []
source_list = []
sat_inputs_2, sat_source_list = doenut.add_higher_order_terms(
    new_inputs,
    add_squares=True,
    add_interactions=True,
    column_list=[])
```

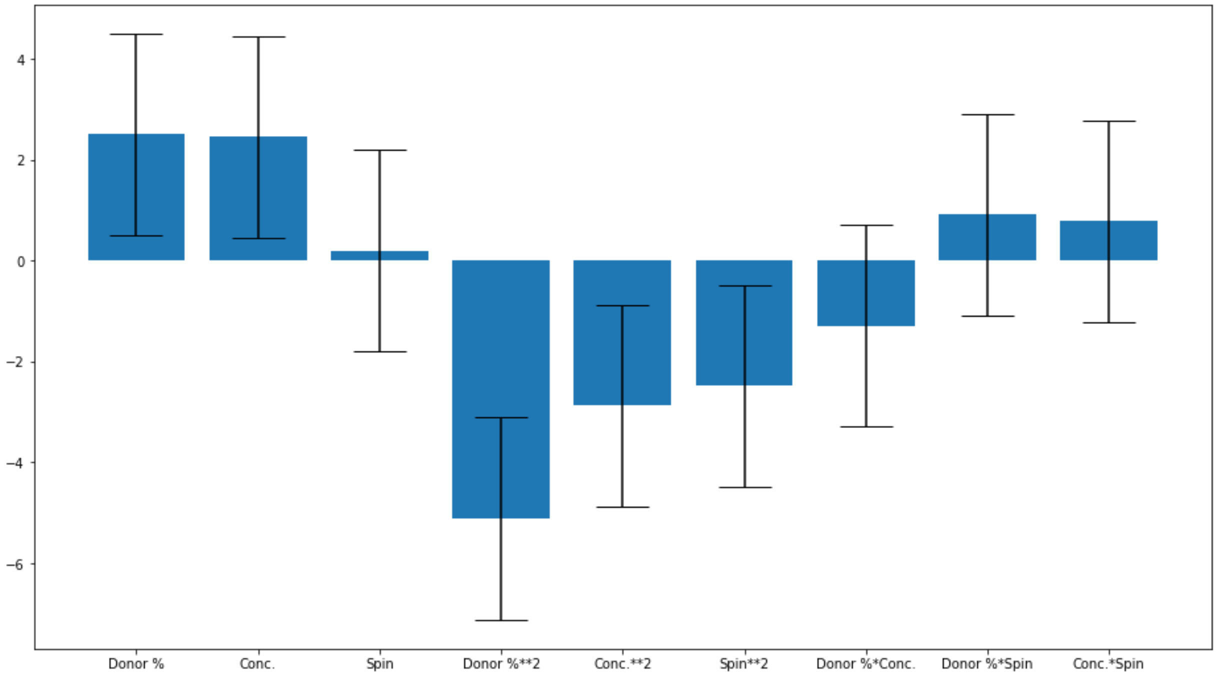
```
Input array has columns ['Donor %', 'Conc.', 'Spin']
Adding square terms:
Donor %**2
Conc.**2
Spin**2
Adding interaction terms:
Donor %*Conc.
Donor %*Spin
Conc.*Spin
```

Saturated model: 9 terms

In [16]:
```
input_selector = [0, 1, 2,
                 3,4,5,
                 6,7,8]


scaled_model, R2, temp_tuple, _ =doenut.tune_model(
                sat_inputs_2,
                new_responses,
                input_selector=input_selector,
                response_selector=[0]
              )

new_model, predictions, ground_truth, coeffs, R2s, R2, Q2= temp_tuple
```

```
Input terms are ['Donor %', 'Conc.', 'Spin', 'Donor %**2', 'Conc.**2', 'Spin**2', 'D
onor %*Conc.', 'Donor %*Spin', 'Conc.*Spin']
Input Responses are ['PCE']

Selected Response is PCE
Selected input terms:   ['Donor %', 'Conc.', 'Spin', 'Donor %**2', 'Conc.**2', 'Spin
**2', 'Donor %*Conc.', 'Donor %*Spin', 'Conc.*Spin']
Have found no replicates
Input data is 27 points long
We are using 27 data points
Left out data point 0:  R2 = 0.851      Ave. Error = -1.01
Left out data point 1:  R2 = 0.909      Ave. Error = -2.8
Left out data point 2:  R2 = 0.759      Ave. Error = 13.7
Left out data point 3:  R2 = 0.896      Ave. Error = -2.5
Left out data point 4:  R2 = 0.91       Ave. Error = 1.77
Left out data point 5:  R2 = 0.919      Ave. Error = -2.62
Left out data point 6:  R2 = 0.926      Ave. Error = 5.98
Left out data point 7:  R2 = 0.891      Ave. Error = 0.493
Left out data point 8:  R2 = 0.911      Ave. Error = -3.07
Left out data point 9:  R2 = 0.896      Ave. Error = -0.901
Left out data point 10: R2 = 0.926      Ave. Error = -3.49
Left out data point 11: R2 = 0.901      Ave. Error = -2.85
Left out data point 12: R2 = 0.89       Ave. Error = 1.02
Left out data point 13: R2 = 0.906      Ave. Error = 2.29
Left out data point 14: R2 = 0.889      Ave. Error = 0.127
Left out data point 0:  R2 = 0.851      Ave. Error = 0.722
Left out data point 1:  R2 = 0.909      Ave. Error = 0.469
Left out data point 2:  R2 = 0.759      Ave. Error = 1.13
Left out data point 3:  R2 = 0.896      Ave. Error = -0.258
Left out data point 4:  R2 = 0.91       Ave. Error = 1.92
Left out data point 5:  R2 = 0.919      Ave. Error = -0.342
Left out data point 6:  R2 = 0.926      Ave. Error = -0.0549
Left out data point 7:  R2 = 0.891      Ave. Error = 0.741
Left out data point 8:  R2 = 0.911      Ave. Error = 0.793
Left out data point 9:  R2 = 0.896      Ave. Error = 0.156
Left out data point 10: R2 = 0.926      Ave. Error = 1.35
Left out data point 11: R2 = 0.901      Ave. Error = 0.614
R2 overall is 0.89
Mean of test set: 4.746915925925926
Mean being used: 4.746915925925926
Sum of squares of the residuals (explained variance) is 294.06908219945797
Sum of squares total (total variance) is 206.3168644580518
Q2 is -0.425
Response PCE R2 is 0.896
Input selector was [0, 1, 2, 3, 4, 5, 6, 7, 8]
Input_selector was: [0, 1, 2, 3, 4, 5, 6, 7, 8]
Average coefficients are: [ 2.50008245  2.4484955   0.19604689 -5.11218527 -2.877894
7  -2.48480065
 -1.28965495  0.90939463  0.78220542]
Coefficient labels are: ['Donor %', 'Conc.', 'Spin', 'Donor %**2', 'Conc.**2', 'Spin
**2', 'Donor %*Conc.', 'Donor %*Spin', 'Conc.*Spin']
```





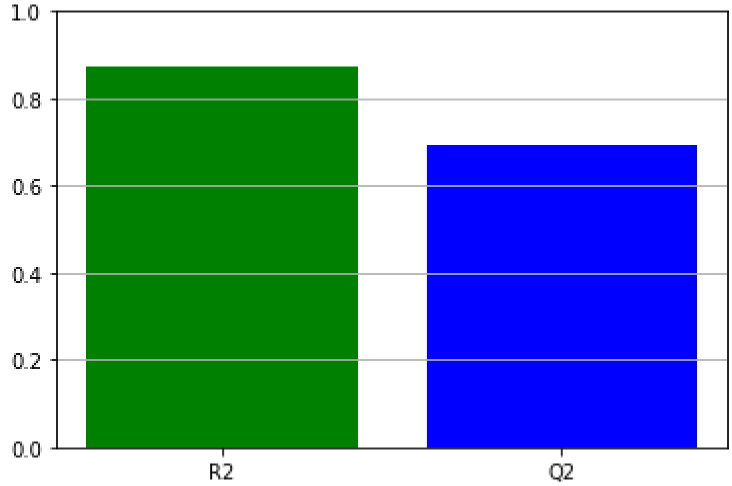
Optimised parsimonious model

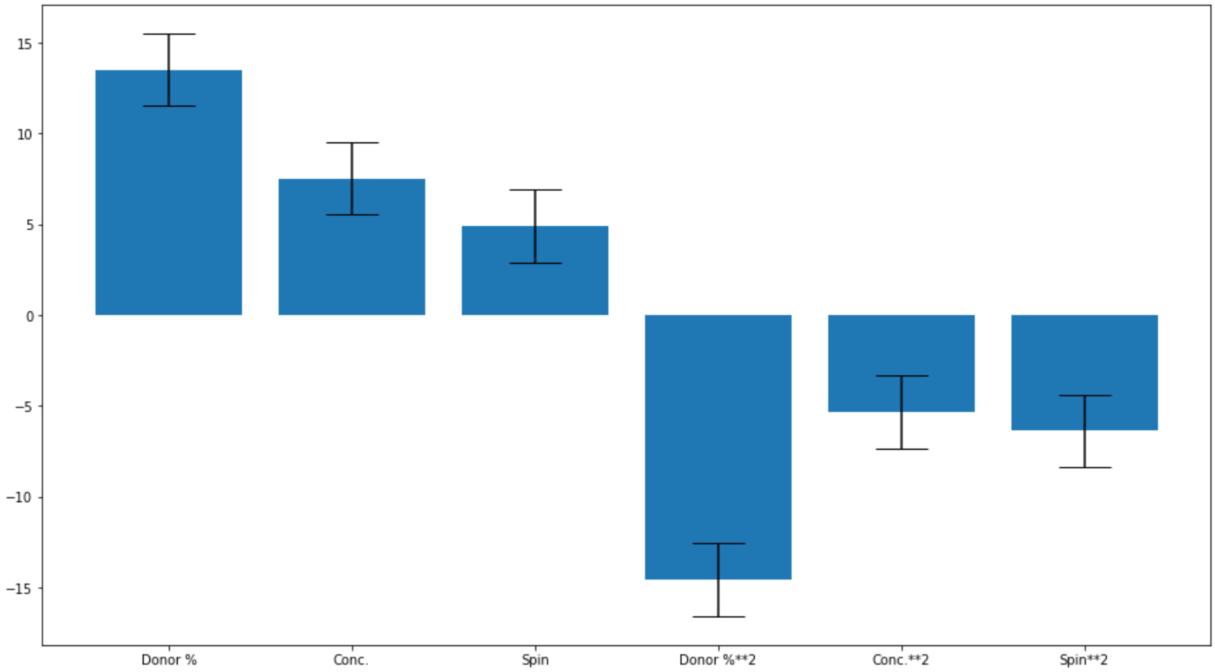
In [17]:

```
input_selector = [0, 1, 2,
                  3, 4, 5]


scaled_model, R2, temp_tuple, _ =doenut.tune_model(
                sat_inputs_2,
                new_responses,
                input_selector=input_selector,
                response_selector=[0]
              )

new_model, predictions, ground_truth, coeffs, R2s, R2, Q2= temp_tuple
```

```
Input terms are ['Donor %', 'Conc.', 'Spin', 'Donor %**2', 'Conc.**2', 'Spin**2', 'D
onor %*Conc.', 'Donor %*Spin', 'Conc.*Spin']
Input Responses are ['PCE']

Selected Response is PCE
Selected input terms:   ['Donor %', 'Conc.', 'Spin', 'Donor %**2', 'Conc.**2', 'Spin
**2']
Have found no replicates
Input data is 27 points long
We are using 27 data points
Left out data point 0:  R2 = 0.856      Ave. Error = -1.01
Left out data point 1:  R2 = 0.869      Ave. Error = -0.921
Left out data point 2:  R2 = 0.867      Ave. Error = 2.28
Left out data point 3:  R2 = 0.889      Ave. Error = -3.22
Left out data point 4:  R2 = 0.891      Ave. Error = 2.07
Left out data point 5:  R2 = 0.892      Ave. Error = -2.53
Left out data point 6:  R2 = 0.855      Ave. Error = 0.63
Left out data point 7:  R2 = 0.865      Ave. Error = 0.597
Left out data point 8:  R2 = 0.874      Ave. Error = -1.67
Left out data point 9:  R2 = 0.869      Ave. Error = -0.689
Left out data point 10: R2 = 0.921      Ave. Error = -3.98
Left out data point 11: R2 = 0.864      Ave. Error = -0.207
Left out data point 12: R2 = 0.87       Ave. Error = 1.83
Left out data point 13: R2 = 0.879      Ave. Error = 1.75
Left out data point 14: R2 = 0.872      Ave. Error = -0.627
Left out data point 0:  R2 = 0.856      Ave. Error = 0.137
Left out data point 1:  R2 = 0.869      Ave. Error = 0.216
Left out data point 2:  R2 = 0.867      Ave. Error = 0.46
```

```
Left out data point 3:  R2 = 0.889      Ave. Error = -0.326
Left out data point 4:  R2 = 0.891      Ave. Error = 1.96
Left out data point 5:  R2 = 0.892      Ave. Error = -0.415
Left out data point 6:  R2 = 0.855      Ave. Error = 0.37
Left out data point 7:  R2 = 0.865      Ave. Error = 0.799
Left out data point 8:  R2 = 0.874      Ave. Error = 0.909
Left out data point 9:  R2 = 0.869      Ave. Error = 0.0224
Left out data point 10: R2 = 0.921      Ave. Error = 1.37
Left out data point 11: R2 = 0.864      Ave. Error = 0.499
R2 overall is 0.871
Mean of test set: 4.746915925925926
Mean being used: 4.746915925925926
Sum of squares of the residuals (explained variance) is 62.93998023442563
Sum of squares total (total variance) is 206.3168644580518
Q2 is 0.695
Response PCE R2 is 0.871
Input selector was [0, 1, 2, 3, 4, 5]
Input_selector was: [0, 1, 2, 3, 4, 5]
Average coefficients are: [ 13.51406635   7.52612545   4.90682899 -14.54539532  -5.3
2766778
  -6.36236656]
Coefficient labels are: ['Donor %', 'Conc.', 'Spin', 'Donor %**2', 'Conc.**2', 'Spin
**2']
```





In [18]:
```
27-1-9
```

Out[18]: 17

No. of terms	DoF	term removed	factor removed	R^2	Q^2
9	17			0.89	-0.425
8	18	8	Conc.*Spin	0.887.	0.44
7	19	7	Donor*Spin	0.88	0.535
6	20	6	Donor* Conc	0.871	0.695

In [ ]:

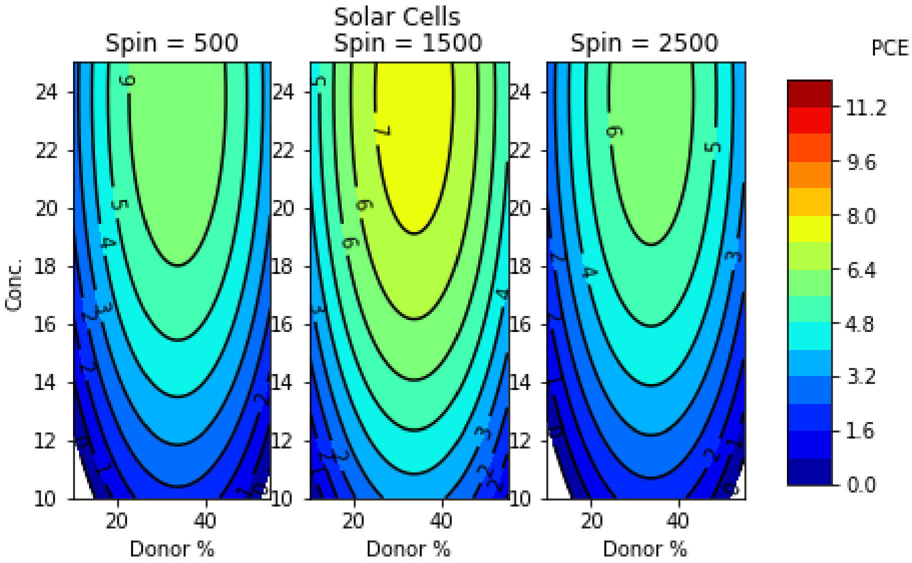
Task 5: Optimising the devices. Using the best model that you have trained (as measured by Q2), find some conditions to optimise the devices.

Task 5. Method 1: Plot a 4-D contour plot and read the values off:

In [19]:

```
#'Donor %', 'Conc.', 'Spin'

n_points = 60


def my_function(df_1):
    ## Put the two main factors that you're not plotting here
    ## set them to sensible constant values

    df_1['Donor %**2'] = df_1['Donor %']*df_1['Donor %']
    df_1['Conc.**2'] = df_1['Conc.']*df_1['Conc.']
    df_1['Spin**2'] = df_1['Spin'] * df_1['Spin']

    return df_1



c_key ='Spin'
y_key='Conc.'
x_key="Donor %"

doenut.four_D_contour_plot(
    unscaled_model=scaled_model,
    x_key=x_key,
    y_key=y_key,
    c_key=c_key,
    x_limits=[inputs[x_key].min(),inputs[x_key].max()],
    y_limits=[inputs[y_key].min(),inputs[y_key].max()],
    constants=[500,1500,2500],
    n_points=60,
    my_function=my_function,
    input_selector=[],
    fig_label='Solar Cells',
    x_label=x_key,
    y_label=y_key,
    constant_label=c_key,
    z_label = 'PCE',
    cmap='jet',
    num_of_z_levels=16,
    z_limits=[0,12])
```

```
<Figure size 1440x864 with 0 Axes>
```



Task 5. Method 2. Run the model on the input values:

In [20]:
```
question_5p2=pd.DataFrame({'A':{'Donor %': 20, 'Conc.': 12, 'Spin':  500},
 'B':{'Donor %': 40, 'Conc.': 16, 'Spin':  1500},
 'C':{'Donor %': 35, 'Conc.': 22, 'Spin':  1500},
 'D':{'Donor %': 45, 'Conc.': 18, 'Spin':  2500},
 'E':{'Donor %': 20, 'Conc.': 17, 'Spin':  2500}}).T
```

In [21]:
```
question_5p2
```

Out[21]:

	Donor %	Conc.	Spin
A	20	12	500
B	40	16	1500
C	35	22	1500
D	45	18	2500
E	20	17	2500

In [22]:
```
question_5p2.index
```

Out[22]:
```
Index(['A', 'B', 'C', 'D', 'E'], dtype='object')
```

In [23]:
```
sat_source_list = []
source_list = []
sat_inputs_q5, sat_source_list = doenut.add_higher_order_terms(
    question_5p2,
    add_squares=True,
    add_interactions=True,
    column_list=[],
    verbose=False)

results, _ =doenut.predict_from_model(
    scaled_model,
    sat_inputs_q5,
    input_selector)
letters = [x for x in question_5p2.index]
[print(f"{letters[i]}:\t{results[i]}") for i in range(len(letters))];
```

```
A:	2.105294443842542
B:	6.111167642320282
```

```
C:      7.610495288984239
D:      4.653001005905132
E:      3.969517548689133
```

Answer C is above 7.

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```