機械学習を使った調達業務の価格査定について

未来調達研究所株式会社　坂口孝則（info@future-procurement.com）15th Apr 2017

この報告書は機械学習を活用することによって調達業務の価格査定が進化するかを述べたものである。結論から述べれば、機械学習を活用することにより、従来手法以上の精度で価格を予想できる。それと同時に、価格決定理由について、調達担当者自身が説明できない、という大きな問題を残すものとなった。

結論だけ知りたい方は、節「2.」～「5.」を飛ばして、「6.まとめ」からお読みになっても問題がない。

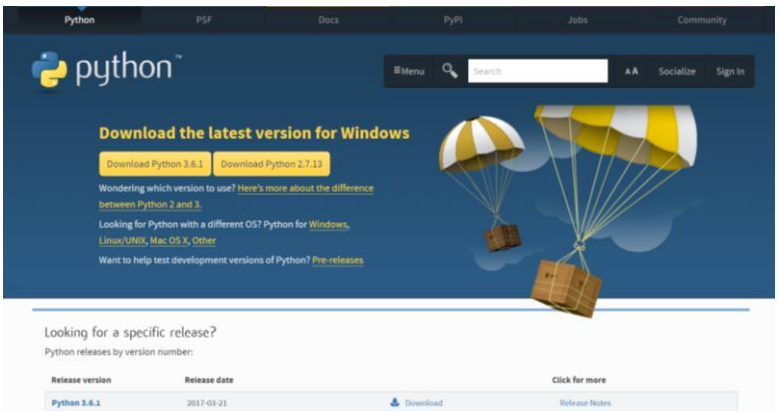
1.はじめに

もともと機械学習とは、統計処理を活用したものだ。何かと何かのデータ（説明変数）と、何かのデータ（目的変数）に法則性を見つける。さらにデータ数が増えるほど、その法則が進化し、精度よく未知の目的変数を予想できるようになる。

たとえば、調達業務では調達品の価格査定がある。一つの対象物を調達する際に、さまざまな仕様情報から価格を査定する。そのときに、それらの仕様情報から価格を積み上げる方法や、あるいは特定仕様との相関を調べる方法などが考案されてきた。今回は、機械学習を活用し、仕様情報から価格を査定できるかを試行した。また同時に、それが旧来の方法と比べて優位性があるのかも確認した。

2.環境

Python(https://www.python.org/downloads/)を使用する。この言語は統計処理や機械学習のライブラリを容易に活用できるためである。



そして、この Python では多くの開発者が無料でパッケージ（アルゴリズム）を公開しているため、おなじく活用が容易となっている（ https://pypi.python.org/pypi ）。

3.分析対象（サンプル）

今回は次のような調達品の仕様データを用いる。これは金属加工品の調達実績である。行は 45 となっており、これは 45 の調達実績を分析していることを指す。45 回、金属加工品を調達しており、調達実績について、各仕様を表にした。

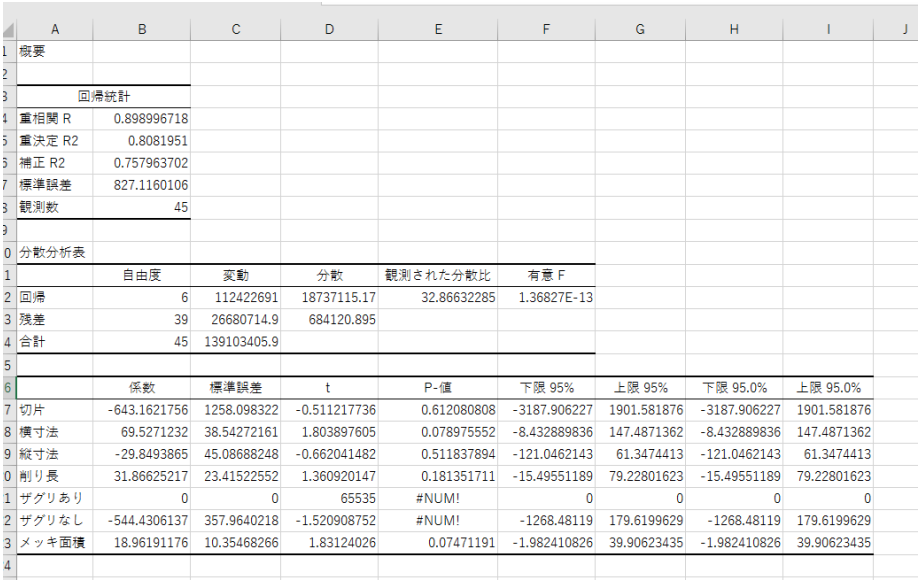
横寸法	縦寸法	削り長	ザグリあり	ザグリなし	メッキ面積	単価
28	44	25	0	1	22.6	442
33	51	30	0	1	35.6	881
28	39	17	0	1	9.8	352
28	39	17	0	1	9.8	335
22	30	13	0	1	4.2	525
38	52	32	0	1	31.7	400
56	65	20	0	1	17.1	2,538
67	78	24	0	1	30	2,896
77	88	25	0	1	35.6	3,735
87	100	27	0	1	51.5	3,948
95	110	34	0	1	82.1	4,767
105	120	34	0	1	90.1	5,287
130	150	40	0	1	175.8	8,929
26	39	30	0	1	19.9	1,206
28	39	30	0	1	17.4	926
33	47	32	0	1	28.1	1,760
35	47	30	0	1	23.2	992
38	52	36	0	1	35.6	1,886
38	54	40	0	1	46.2	2,296
38	56	40	0	1	53.1	2,446
48	62	40	0	1	48.4	2,317
22	39	30	1	0	24.4	2,515
30	47	30	1	0	30.8	2,593
32	52	36	1	0	47.5	2,624
32	38	40	1	0	13.2	526
40	62	40	1	0	70.5	4,261
45	74	49	1	0	132.7	6,300
35	55	27	1	0	38.2	1,065
59	72	30	1	0	40.1	2,096
55	80	34	1	0	90.1	2,751
55	80	34	1	0	90.1	4,208
32	52	27	1	0	35.6	947
22	39	30	0	1	24.4	1,316
25	60	20	1	0	46.7	1,065
30	55	28	1	0	46.7	2,187
48	78	36	1	0	106.8	2,602
56	90	36	1	0	140.3	3,309
60	97	38	1	0	173.3	3,914
36	55	28	0	1	38	1,132
28	39	17	0	1	9.8	341
48	62	30	0	1	36.3	1,082
58	72	30	0	1	42.9	1,130
35	47	30	0	1	23.2	598
40	52	36	0	1	31.2	1,093
40	52	36	0	1	31.2	1,298

4.伝統的な価格査定アプローチ

ここでまず伝統的な価格査定のアプローチを用いる。エクセルのデータ分析から、回帰分析を実施する。

横寸法	縦寸法	削り長	ザグリあり	ザグリなし	メッキ面積	単価
28	44	25	0	1	22.608	442
33	51	30	0	1	35.6076	881
28	39	17	0	1	9.835265	352
28	39	17	0	1	9.835265	335
22	30	13	0	1	4.24528	525
38	52	32	0	1	31.6512	400
56	65	20	0	1	17.0973	2538
67	78	24	0	1	30.0498	2896
77	88	25	0	1	35.619375	3735
87	100	27	0	1	51.525045	3948
95	110	34	0	1	82.07175	4767
105	120	34	0	1	90.07875	5287
130	150	40	0	1	175.84	8929
26	39	30	0	1	19.89975	1206
28	39	30	0	1	17.35635	926
33	47	32	0	1	28.1344	1760

Y は価格を設定し、そして X には仕様データを一括指定する。これらの統計的根拠については書籍（たとえば「"とびきりやさしい"ビジネス統計入門」 https://www.amazon.co.jp/dp/4526072907 など）を参照のこと。重要なことは、結果として、下のような分析を導くことである。



概要

回帰統計	
重相関 R	0.898996718
重決定 R2	0.8081951
補正 R2	0.757963702
標準誤差	827.1160106
観測数	45

分散分析表

	自由度	変動	分散	観測された分散比	有意 F
回帰	6	112422691	18737115.17	32.86632285	1.36827E-13
残差	39	26680714.9	684120.895		
合計	45	139103405.9			

	係数	標準誤差	t	P-値	下限 95%	上限 95%	下限 95.0%	上限 95.0%
切片	-643.1621756	1258.098322	-0.511217736	0.612080808	-3187.906227	1901.581876	-3187.906227	1901.581876
横寸法	69.5271232	38.54272161	1.803897605	0.078975552	-8.432889836	147.4871362	-8.432889836	147.4871362
縦寸法	-29.8493865	45.08688248	-0.662041482	0.511837894	-121.0462143	61.3474413	-121.0462143	61.3474413
削り長	31.86625217	23.41522552	1.360920147	0.181351711	-15.49551189	79.22801623	-15.49551189	79.22801623
ザグリあり	0	0	65535	#NUM!	0	0	0	0
ザグリなし	-544.4306137	357.9640218	-1.520908752	#NUM!	-1268.48119	179.6199629	-1268.48119	179.6199629
メッキ面積	18.96191176	10.35468266	1.83124026	0.07471191	-1.982410826	39.90623435	-1.982410826	39.90623435

そこで、一例を示す。このデータ分析の元リストには入っていないものの、このような仕様の製品がある。

- 横寸法：30
- 縦寸法：47
- 削り長：23
- ザグリあり：1
- ザグリなし：0
- メッキ面積：24

そしてこの調達品の実績価格は「931 円」であった。この 931 円にどれほど近づけるかを試算する。

4-1.伝統的な価格査定アプローチ（単回帰コストドライバー分析）

そこで、まずは単回帰コストドライバー分析を行う。詳細については書籍（たとえば、「調達・購買の教科書」 https://www.amazon.co.jp/dp/4526070084 など ）を参照のこと。ここでは、価格にもっとも影響を及ぼすメッキ面積を採用する（なぜならば、前述の回帰分析の結果から t 値がもっとも高い）。

よって、予想したい下記の製品価格について、

- 横寸法：30
- 縦寸法：47
- 削り長：23
- ザグリあり：1
- ザグリなし：0
- メッキ面積：24

メッキ面積 24 のみを使用し、価格を類推する。forecast 関数によって、1,332.5 円と予想できる。forecast 関数では、既知の X/Y を指定することになっているから、それぞれメッキ面積の 45 実績、単価の 45 実績をあてはめ、そして知りたい値は 24 を入力すればよい。

	A	B	C	D	E	F	G	H	I
1	横寸法	縦寸法	削り長	ザグリあり	ザグリなし	メッキ面積	単価		1,332.5
2	28	44	25	0	1	22.6	442		
3	33	51	30	0	1	35.6	881		
4	28	39	17	0	1	9.8	352		
5	28	39	17	0	1	9.8	335		
6	22	30	13	0	1	4.2	525		
7	38	52	32	0	1	31.7	400		
8	56	65	20	0	1	17.1	2,538		
9	67	78	24	0	1	30	2,896		
10	77	88	25	0	1	35.6	3,735		
11	87	100	27	0	1	51.5	3,948		
12	95	110	34	0	1	82.1	4,767		
13	105	120	34	0	1	90.1	5,287		
14	130	150	40	0	1	175.8	8,929		
15	26	39	30	0	1	19.9	1,206		
16	28	39	30	0	1	17.4	926		
17	33	47	32	0	1	28.1	1,760		

4-1.伝統的な価格査定アプローチ（重回帰コストドライバー分析）

次に、複数のコストドライバーによる分析、重回帰分析を行う。この場合、すでに回帰分析を終了しているから、結果の係数が使用できる。

結果、次の図に計算結果を書いているとおり、1,227.7 円と予想できる。

- 実際の値：931 円
- 単回帰コストドライバー分析：1,332.5 円
- 重回帰コストドライバー分析：1,227.7 円

概要

回帰統計	
重相関 R	0.898996718
重決定 R2	0.8081951
補正 R2	0.757963702
標準誤差	827.1160106
観測数	45

分散分析表

	自由度	変動	分散	観測された分散比	有意 F
回帰	6	112422691	18737115.17	32.86632285	1.36827E-13
残差	39	26680714.9	684120.895		
合計	45	139103405.9			

	係数	標準誤差	t	P-値	下限 95%	上限 95%	下限 95.0%	上限 95.0%
切片	-643.1621756	1258.098322	-0.511217736	0.612080808	-3187.906227	1901.581876	-3187.906227	1901.581876
横寸法	69.5271232	38.54272161	1.803897605	0.078975552	-8.432889836	147.4871362	-8.432889836	147.4871362
縦寸法	-29.8493865	45.08688248	-0.662041482	0.511837894	-121.0462143	61.3474413	-121.0462143	61.3474413
削り長	31.86625217	23.41522552	1.360920147	0.181351711	-15.49551189	79.22801623	-15.49551189	79.22801623
ザグリあり	0	0	65535	#NUM!	0	0	0	0
ザグリなし	-544.4306137	357.9640218	-1.520908752	#NUM!	-1268.48119	179.6199629	-1268.48119	179.6199629
メッキ面積	18.96191176	10.35468266	1.83124026	0.07471191	-1.982410826	39.90623435	-1.982410826	39.90623435

試算	単価推計
1	-643.2
30	2,085.8
47	-1,402.9
23	732.9
1	0.0
0	0.0
24	455.1
	1,227.7

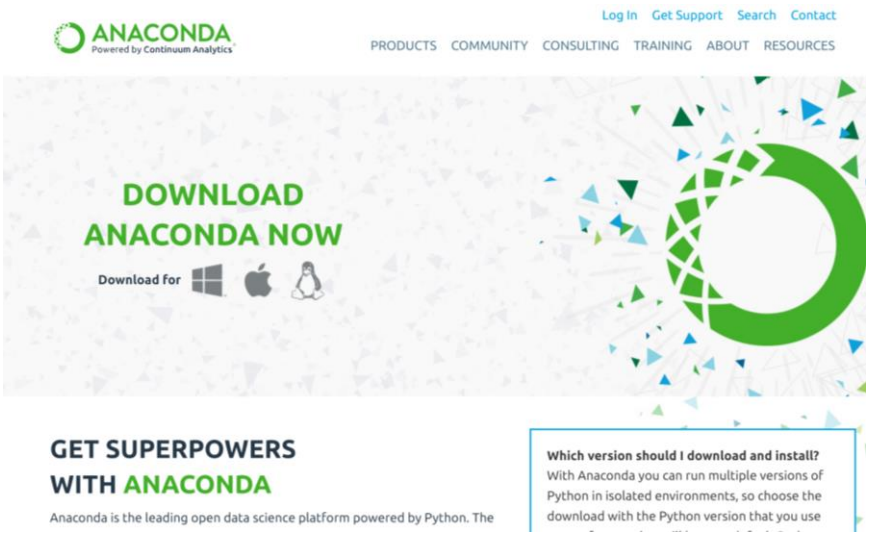
この結果を見ると、おおむね重回帰コストドライバー分析のほうが実際の値に近づいていることがわかる。

5.Python による価格推計

分析には末尾に載せたコードを使用する。

```
File  Edit  Format  Run  Options  Window  Help
from sklearn import cross_validation, svm, metrics

# 価格データ(CSV)を読みこむ
price_csv= []
with open("pricelist3.csv", "r", encoding="utf-8") as fp:
    no = 0
    for line in fp:
        line = line.strip()
        cols = line.split(";")
        price_csv.append(cols)

# 先頭行は名称を記載しているため削除する
price_csv = price_csv[1:]

# 価格データに載っている情報を数値に変換
labels = []
data = []
for cols in price_csv:
    cols = list(map(lambda n: float(n), cols))
    # ここで価格予想を修正した
    grade = int(cols[6]) # 末尾が価格データ
    #if grade == 9: grade = 8 # 少なすぎるサンプルを調整
    #if grade < 4 : grade = 5
    labels.append(grade)
    data.append( cols[0:6] ) # この末尾が価格データ

# 訓練用データとテスト用データに分ける
data_train, data_test, label_train, label_test = ¥
    cross_validation.train_test_split(data, labels)

# アルゴリズムのうちランダムフォレストを利用
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier()

#リニアモデル（現在は使っていないが、使う際には#を外す）
#from sklearn import linear_model
#clf = linear_model.LinearRegression()

#SVM使うなら（現在は使っていないが、使う際には#を外す）
#clf = svm.LinearSVC()
clf.fit(data_train, label_train)

# サンプルを使用し予測実施
predict = clf.predict(data_test)
total = ok = 0
for idx,pre in enumerate(predict):
        # pre = predict[idx]     # 予測したラベル
        answer = label_test[idx] # 正解ラベル
        total += 1
        # 価格をぴったり当てられることはありえないので、前後数パーセントを許容
        # 下のサンプルでは±10%を許容した
        if (int(answer) * 0.90 ) <= pre <= (int(answer) * 1.10 ):
            ok += 1
print("正解率=", ok, "/", total, "=", ok/total)
```

なお、このライブラリは、anaconda（ https://www.continuum.io/downloads ）をダウンロードして使用した。これをダウンロードすることで、scikit-learn 等の分析アルゴリズムが使えるからである。



なお、本文はプログラミングの解説文ではないため、コードについては最小限の説明でとどめる。ただ、読者が実践する場合は、以下に注意されたい。

- 使用する csv ファイルは単価情報がもっとも右にくるものとし、そして行の数に応じて、「grade = int(cols[6]) # 末尾が価格データ」の数字箇所を変更すること。サンプルでは仕様が 6 列並んでおり、7 列目が単価データになっている。左端を 0 と数えるため、単価列は 6 となっている。
- アルゴリズムはランダムフォレストを使用している。ただ、その他、リニアの線形モデル、SVM などもあらかじめ組み込んでいるため、それらのモデルを活用する場合は、コードの「#」を削除のうえ、逆にランダムフォレスト箇所に「#」付与すること
- 正解率の計算については、「if (int(answer) * 0.90) <= pre <= (int(answer) * 1.10):」としている。これは、機会が学習し求めた結果、10 円だとする。そのとき、ほんとうの調達価格は 9～11 円だとしても、10%ていどのズレは正解とするものである。もちろん、厳し目に試算したい場合は、上記、0.90、1.10 を修正すると良い
- 下から二行目に「n=clf.predict([30, 47, 23, 1, 0, 24])」と書いている。これは前述のとおり、予想したい製品仕様である。ここを書き換えると、新たな数字で予想が可能となる。

そこで結果を確認してみる。

このコードでは、ランダムにデータをわける。いっぽうのデータ群は答え（単価）をあらかじめ機械に教え、単価の法則性を学習させる。そしてもういっぽうのデータ群に、その法則を当てはめてみて、どれだけ正しく計算できたかを確認する。今回の場合は、精度が±10%であれば正解となる。

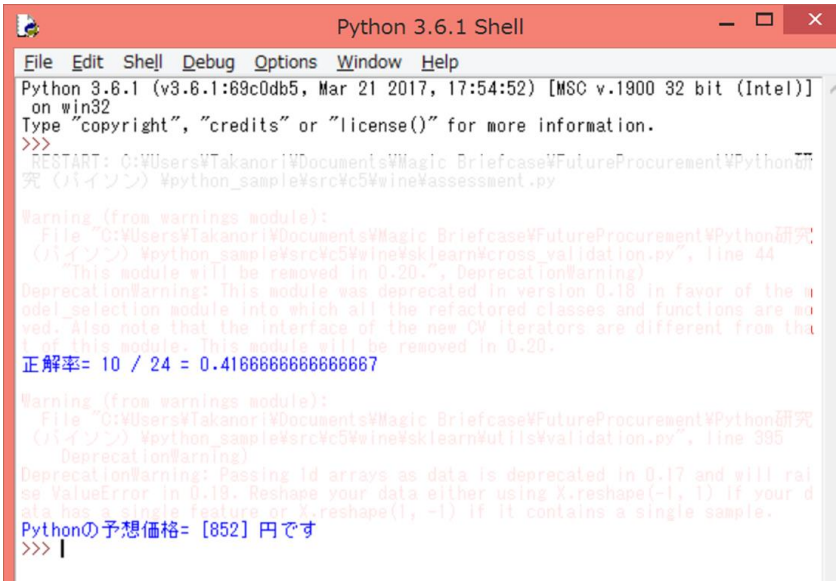
ランダムがゆえに、かならずおなじ正解率になるわけではない。ただ図示の例では、正解率が≒33%となった。予想価格は 1,065 円とある。

- 実際の値：931 円
- 単回帰コストドライバー分析：1,332.5 円
- 重回帰コストドライバー分析：1,227.7 円
- Python（scikit-learn）：1,065 円

こういう結果となった。

5-1.Python による価格推計（45 個→93 個）

次に、45 の調達実績だったところ、さらに事例を増やし 93 とした。これで再度、コードを動かしてみる。



すると、結果としては正答率が≒42%に向上し、予想価格は 852 円となった。

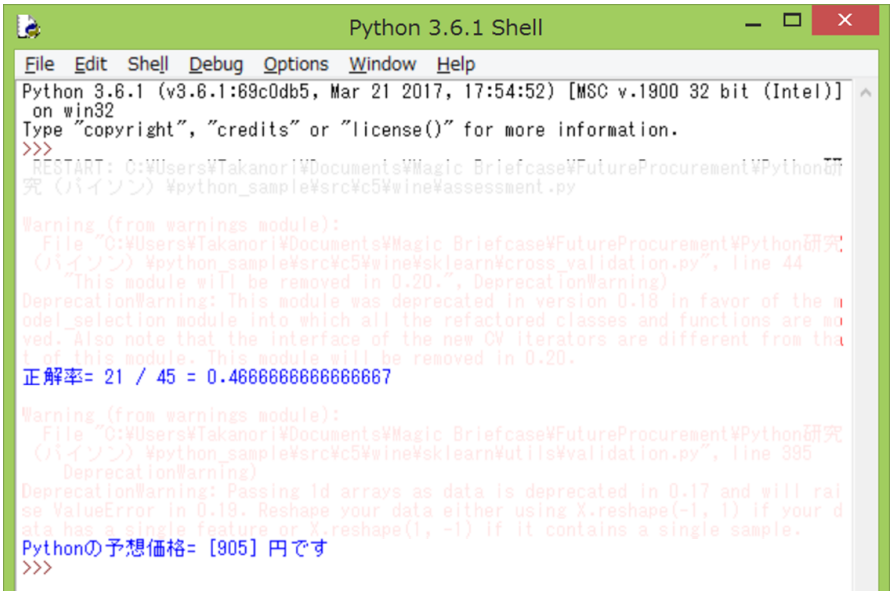
比較してみる。

- 実際の値：931 円
- 単回帰コストドライバー分析：1,332.5 円
- 重回帰コストドライバー分析：1,227.7 円
- Python（scikit-learn）：1,065 円
- Python（scikit-learn）データ増：852 円

5-2.Python による価格推計（93 個→178 個）

さらに事例を増やし 178 とした。これで再度、コードを動かしてみる。このように 3 段階で実施したのは、意図したものではない。実際の調達・購買業務において、サンプルを用意するのに時間を要したのが大きい。さらに、分析する前にはどれだけの成果があがるかわからず、少量から開始した、という実務的な問題がある。

次が結果となる。



正解率は≒46%となり、予想価格は 905 円となった。

比較を行う。

- 実際の値：931 円
- 単回帰コストドライバー分析：1,332.5 円
- 重回帰コストドライバー分析：1,227.7 円
- Python（scikit-learn）：1,065 円
- Python（scikit-learn）データ増：852 円
- Python（scikit-learn）データ増＋さらに増：905 円

なお、データが増加しているため、重回帰分析を再度実施する必要がある。ただ、データ 178 個で重回帰分析を実施しても結果はさほど変わらないと補足しておく。

6.まとめ

あらためて抜粋して比較しておく。

- 実際の値：931 円（←この値をもっとも正確に予想できるモデルを模索した）
- 重回帰コストドライバー分析：1,227.7 円（←調達査定で実際に使われるモデル）
- Python（scikit-learn）：905 円（←今回、機械学習により導いたモデル）

今回はあくまで一例の価格予想しか載せていない。ただし、おおむね他の価格査定においても機械学習における良好な結果が出ている。これまでの価格推計のアプローチであるコストドライバー分析にくらべても、近似した値が確認できる。

当然ではあるものの、データを分析する量が重要となる。数データよりも、莫大なデータのほうが学習結果は良好となる。企業で調達実績を多く収集できる場合は、機械学習による価格査定はきわめて大きなツールとなる可能性がある。

7.問題点

超データの観点から問題点を述べておく。この機械学習による査定の問題点は、調達担当者がそのロジックを理解できないところにある。たとえば、ランダムフォレストが最終的に導いた単価算出式がわからない

まま、単価が導かれる（実際はモデル化されているものの、その高度数学を現場で活用するのは現実的ではない意味で「ランダムフォレストが最終的に導いた単価算出式がわからない」と書いた)。

本来、価格査定は「因果」を起点とする。つまり、仕様 A と仕様 B と仕様 C を積み上げて、単価が試算される。しかし、機械学習による査定は「相関」的査定といえるだろう。「理由はさておき、とりあえずこの単価と査定できる」という態度にたいする“態度”が問われているように思われる。

とはいえ、おそらく製品による使い分けによって大きく活用余地があると思われる。自社での展開をお考えの方は未来調達研究所株式会社までご連絡をお願いしたい。

info@future-procurement.com

【参考】シミュレーションコード

```
from sklearn import cross_validation, svm, metrics

# 価格データ(CSV)を読みこむ
price_csv= []
with open("pricelist3.csv", "r", encoding="utf-8") as fp:
    no = 0
    for line in fp:
        line = line.strip()
        cols = line.split(";")
        price_csv.append(cols)

# 先頭行は名称を記載しているため削除する
price_csv = price_csv[1:]

# 価格データに載っている情報を数値に変換
labels = []
data = []
for cols in price_csv:
    cols = list(map(lambda n: float(n), cols))
    # ここで価格予想を修正した
    grade = int(cols[6]) # 末尾が価格データ
    labels.append(grade)
    data.append( cols[0:6] ) # この末尾が価格データ

# 訓練用データとテスト用データに分ける
data_train, data_test, label_train, label_test = ¥
    cross_validation.train_test_split(data, labels)

# アルゴリズムのうちランダムフォレストを利用
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier()

#リニアモデル（現在は使っていないが、使う際には#を外す）
#from sklearn import linear_model
#clf = linear_model.LinearRegression()

#SVM 使うなら（現在は使っていないが、使う際には#を外す）
#clf = svm.LinearSVC()
clf.fit(data_train, label_train)

# サンプルを使用し予測実施
predict = clf.predict(data_test)
total = ok = 0
for idx,pre in enumerate(predict):
        # pre = predict[idx]        # 予測したラベル
        answer = label_test[idx] # 正解ラベル
        total += 1
        # 価格をぴったり当てられることはありえないので、前後数パーセントを許容
        # 下のサンプルでは±10%を許容した
        if (int(answer) * 0.90 ) <= pre <= (int(answer) * 1.10 ):
            ok += 1
print("正解率=", ok, "/", total, "=", ok/total)

# 具体的に表に載っていない一つの例で予想させる
n=clf.predict([30, 47, 23, 1, 0, 24])
print("Python の予想価格=", n, "円です")
```

【参考】調達実績 178 個時のデータ

横寸法	縦寸法	削り長	ザグリあり	ザグリなし	メッキ面積	単価
28	44	25	0	1	22.6	442
33	51	30	0	1	35.6	881
28	39	17	0	1	9.8	352
28	39	17	0	1	9.8	335
22	30	13	0	1	4.2	525
38	52	32	0	1	31.7	400
56	65	20	0	1	17.1	2,538
67	78	24	0	1	30.0	2,896
77	88	25	0	1	35.6	3,735
87	100	27	0	1	51.5	3,948
95	110	34	0	1	82.1	4,767
105	120	34	0	1	90.1	5,287
130	150	40	0	1	175.8	8,929
26	39	30	0	1	19.9	1,206
28	39	30	0	1	17.4	926
33	47	32	0	1	28.1	1,760
35	47	30	0	1	23.2	992
38	52	36	0	1	35.6	1,886
38	54	40	0	1	46.2	2,296
38	56	40	0	1	53.1	2,446
48	62	40	0	1	48.4	2,317
22	39	30	1	0	24.4	2,515
30	47	30	1	0	30.8	2,593
32	52	36	1	0	47.5	2,624
32	38	40	1	0	13.2	526
40	62	40	1	0	70.5	4,261
45	74	49	1	0	132.7	6,300
35	55	27	1	0	38.2	1,065
59	72	30	1	0	40.1	2,096
55	80	34	1	0	90.1	2,751
55	80	34	1	0	90.1	4,208
32	52	27	1	0	35.6	947
22	39	30	0	1	24.4	1,316
25	60	20	1	0	46.7	1,065
30	55	28	1	0	46.7	2,187
48	78	36	1	0	106.8	2,602
56	90	36	1	0	140.3	3,309
60	97	38	1	0	173.3	3,914
36	55	28	0	1	38.0	1,132
28	39	17	0	1	9.8	341
48	62	30	0	1	36.3	1,082
58	72	30	0	1	42.9	1,130
35	47	30	0	1	23.2	598
40	52	36	0	1	31.2	1,093
40	52	36	0	1	31.2	1,298
25	40	23	0	1	20.0	398
30	46	27	0	1	32.0	793
25	35	15	0	1	9.0	317
25	35	15	0	1	9.0	302
20	27	12	0	1	4.0	473
34	47	29	0	1	28.0	360
50	59	18	0	1	15.0	2,284
60	70	22	0	1	27.0	2,606
69	79	23	0	1	32.0	3,362
78	90	24	0	1	46.0	3,553
86	99	31	0	1	74.0	4,290
95	108	31	0	1	81.0	4,758
117	135	36	0	1	158.0	8,036
23	35	27	0	1	18.0	1,085
25	35	27	0	1	16.0	833
30	42	29	0	1	25.0	1,584
32	42	27	0	1	21.0	893
34	47	32	0	1	32.0	1,697
34	49	36	0	1	42.0	2,066
34	50	36	0	1	48.0	2,201
43	56	36	0	1	44.0	2,085
20	35	27	1	0	22.0	2,264
27	42	27	1	0	28.0	2,334
29	47	32	1	0	43.0	2,362
29	34	36	1	0	12.0	473
36	56	36	1	0	63.0	3,835
41	67	44	1	0	119.0	5,670
27	42	21	1	0	21.0	838
32	50	24	1	0	34.0	959
53	65	27	1	0	36.0	1,886
50	72	31	1	0	81.0	2,476
50	72	31	1	0	81.0	3,787
29	47	24	1	0	32.0	852
20	35	27	0	1	22.0	1,184
23	54	18	1	0	42.0	959
27	50	25	1	0	42.0	1,968
43	70	32	1	0	96.0	2,342
50	81	32	1	0	126.0	2,978
54	87	34	1	0	156.0	3,523
32	50	25	0	1	34.0	1,019
25	35	15	0	1	9.0	307
43	56	27	0	1	33.0	974
52	65	27	0	1	39.0	1,017
32	42	27	0	1	21.0	538
36	47	32	0	1	28.0	984
36	47	32	0	1	28.0	1,168
23	36	21	0	1	18.0	358
27	41	24	0	1	29.0	714
24	37	21	0	1	19.0	376
28	43	26	0	1	30.0	749
24	33	14	0	1	8.0	299
24	33	14	0	1	8.0	285
19	26	11	0	1	4.0	446
32	44	27	0	1	27.0	340
48	55	17	0	1	15.0	2,157
57	66	20	0	1	26.0	2,462
65	75	21	0	1	30.0	3,175
74	85	23	0	1	44.0	3,356
81	94	29	0	1	70.0	4,052
89	102	29	0	1	77.0	4,494
111	128	34	0	1	149.0	7,590
22	33	26	0	1	17.0	1,025
24	33	26	0	1	15.0	787
28	40	27	0	1	24.0	1,496
30	40	26	0	1	20.0	843
32	44	31	0	1	30.0	1,603
32	46	34	0	1	39.0	1,952
32	48	34	0	1	45.0	2,079
41	53	34	0	1	41.0	1,969
19	33	26	1	0	21.0	2,138
26	40	26	1	0	26.0	2,204
27	44	31	1	0	40.0	2,230
27	32	34	1	0	11.0	447
34	53	34	1	0	60.0	3,622
38	63	42	1	0	113.0	5,355
26	40	20	1	0	20.0	791
30	47	23	1	0	32.0	905
50	61	26	1	0	34.0	1,782
47	68	29	1	0	77.0	2,338
47	68	29	1	0	77.0	3,577
27	44	23	1	0	30.0	805
19	33	26	0	1	21.0	1,119
21	51	17	1	0	40.0	905
26	47	24	1	0	40.0	1,859
41	66	31	1	0	91.0	2,212
48	77	31	1	0	119.0	2,813
51	82	32	1	0	147.0	3,327
31	47	24	0	1	32.0	962
24	33	14	0	1	8.0	290
41	53	26	0	1	31.0	920
49	61	26	0	1	36.0	961
30	40	26	0	1	20.0	508
34	44	31	0	1	27.0	929
34	44	31	0	1	27.0	1,103
21	34	20	0	1	17.0	338
26	39	23	0	1	27.0	674
21	30	13	0	1	8.0	269
21	30	13	0	1	8.0	257
17	23	10	0	1	3.0	402
29	40	25	0	1	24.0	306
43	50	15	0	1	13.0	1,941
51	60	19	0	1	23.0	2,215
59	67	20	0	1	27.0	2,858
66	77	20	0	1	39.0	3,020
73	84	26	0	1	63.0	3,647
81	92	26	0	1	69.0	4,044
99	115	31	0	1	134.0	6,831
20	30	23	0	1	15.0	922
21	30	23	0	1	14.0	708
26	36	25	0	1	21.0	1,346
27	36	23	0	1	18.0	759
29	40	27	0	1	27.0	1,442
29	42	31	0	1	36.0	1,756
29	43	31	0	1	41.0	1,871
37	48	31	0	1	37.0	1,772
17	30	23	1	0	19.0	1,924
23	36	23	1	0	24.0	1,984
25	40	27	1	0	37.0	2,008
25	29	31	1	0	10.0	402
31	48	31	1	0	54.0	3,260
35	57	37	1	0	101.0	4,820
23	36	18	1	0	18.0	712
27	43	20	1	0	29.0	815
45	55	23	1	0	31.0	1,603
43	61	26	1	0	69.0	2,105
43	61	26	1	0	69.0	3,219
25	40	20	1	0	27.0	724
17	30	23	0	1	19.0	1,006
20	46	15	1	0	36.0	815
23	43	21	1	0	36.0	1,673
37	60	27	1	0	82.0	1,991
43	69	27	1	0	107.0	2,531
46	74	29	1	0	133.0	2,995