LP problem, Bakery Horvat VT, profit maximization

May 18, 2021

[9]:
```
import numpy as np
import pandas as pd

from sklearn.neighbors import KNeighborsRegressor
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

from sklearn.model_selection import GridSearchCV

from sklearn.preprocessing import LabelEncoder

%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt

from sklearn.pipeline import Pipeline



import numpy as np
import pandas as pd
import statsmodels.api as sm

from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import SGDRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV

from sklearn.svm import LinearSVC
```

```
from sklearn.svm import SVC
import sklearn.linear_model as skl_lm
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split, LeaveOneOut, KFold,␣
↪StratifiedKFold, cross_val_score
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import GridSearchCV

#scikit-learn packages for machine learning algorithms
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.linear_model import SGDRegressor

from sklearn.preprocessing import scale
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV
from sklearn.linear_model import ElasticNet
from sklearn.metrics import mean_squared_error
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.preprocessing import StandardScaler

from sklearn.metrics import confusion_matrix, classification_report,␣
↪precision_score, roc_curve, roc_auc_score, accuracy_score


from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier
from sklearn.tree import export_text, plot_tree, export_graphviz
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import confusion_matrix, mean_squared_error
from sklearn.model_selection import GridSearchCV

#from dtreeviz.trees import dtreeviz
# install graphviz : pip install graphviz
# download graphviz-2.38 and place it into C:/Program Files

import graphviz
import os
os.environ["PATH"] += os.pathsep + 'C:/Program Files/graphviz-2.38/release/bin/'

import matplotlib.pyplot as plt


from IPython.core.interactiveshell import InteractiveShell
```

```
InteractiveShell.ast_node_interactivity = 'all'
```

[10]:
```
import pulp as p
```

[11]:
```
Lp_prob = p.LpProblem('Profit_maximization', p.LpMaximize)
```

[12]:
```
x1 = p.LpVariable("x1", lowBound = 0)
x2 = p.LpVariable("x2", lowBound = 0)
x3 = p.LpVariable("x3", lowBound = 0)
x4 = p.LpVariable("x4", lowBound = 0)
x5 = p.LpVariable("x5", lowBound = 0)
x6 = p.LpVariable("x6", lowBound = 0)
x7 = p.LpVariable("x7", lowBound = 0)
x8 = p.LpVariable("x8", lowBound = 0)
```

[13]:
```
Lp_prob += 4.71 * x1 + 2.39 * x2 + 2.77 * x3 + 2.39 * x4 + 1.99 * x5 + 5.35 *␣
 ↪x6 + 3.44 * x7 + 5.49 * x8


Lp_prob += 0.5 * x1 + 0.042 * x2 + 0.084 * x3 + 0.055 * x4 + 0.058 * x5 + 0.1 *␣
 ↪x6 + 0.084 * x7 + 0.084 * x8 <= 3450
Lp_prob += 0.3 * x1 + 0.025 * x2 + 0.05 * x3 + 0.03 * x4 + 0.032 * x5 + 0.06 *␣
 ↪x6 + 0.05 * x7 + 0.05 * x8 <= 10000
Lp_prob += 0.01 * x1 + 0.0008 * x2 + 0.0016 * x3 + 0.0008 * x4 + 0.0008 * x5 +␣
 ↪0.003 * x6 + 0.0016 * x7 + 0.0016 * x8 <= 70
Lp_prob += 0 * x1 + 0 * x2 + 0 * x3 + 0.001 * x4 + 0.0008 * x5 + 0 * x6 + 0 *␣
 ↪x7 + 0 * x8 <= 5
Lp_prob += 0.01 * x1 + 0.0008 * x2 + 0.0016 * x3 + 0.001 * x4 + 0.001 * x5 + 0.
 ↪002 * x6 + 0.0016 * x7 + 0.0016 * x8 <= 70
Lp_prob += 0 * x1 + 0.00025 * x2 + 0.0005 * x3 + 0.008 * x4 + 0.007 * x5 + 0.
 ↪0007 * x6 + 0.0005 * x7 + 0.0005 * x8 <= 20
Lp_prob += 0 * x1 + 0.00025 * x2 + 0.0005 * x3 + 0.0006 * x4 + 0.0008 * x5 + 0.
 ↪0005 * x6 + 0.0005 * x7 + 0.0005 * x8 <= 6
Lp_prob += 0 * x1 + 0 * x2 + 0 * x3 + 0.25 * x4 + 0.25 * x5 + 0 * x6 + 0 * x7 +␣
 ↪0 * x8 <= 400
Lp_prob += 0 * x1 + 0 * x2 + 0 * x3 + 0.03 * x4 + 0.04 * x5 + 0 * x6 + 0 * x7 +␣
 ↪0 * x8 <= 55
Lp_prob += 0 * x1 + 0 * x2 + 0 * x3 + 0 * x4 + 0 * x5 + 0.04 * x6 + 0.025 * x7␣
 ↪+ 0 * x8 <= 100
Lp_prob += 0 * x1 + 0 * x2 + 0 * x3 + 0 * x4 + 0 * x5 + 0.04 * x6 + 0.025 * x7␣
 ↪+ 0.025 * x8 <= 120
Lp_prob += 0 * x1 + 0 * x2 + 0 * x3 + 0 * x4 + 0 * x5 + 0.025 * x6 + 0 * x7 + 0␣
 ↪* x8 <= 40
Lp_prob += 0 * x1 + 0 * x2 + 0 * x3 + 0 * x4 + 0 * x5 + 0.0002 * x6 + 0 * x7 +␣
 ↪0 * x8 <= 0.5
Lp_prob += 0 * x1 + 0 * x2 + 0 * x3 + 0 * x4 + 0 * x5 + 0 * x6 + 0 * x7 + 0.04␣
 ↪* x8 <= 40
```

```
Lp_prob += 0 * x1 + 0 * x2 + 0 * x3 + 0 * x4 + 0 * x5 + 0 * x6 + 0 * x7 + 0.015␣
 ↪* x8 <= 15
```

[14]:
```
print(Lp_prob)
```

```
Profit_maximization:
MAXIMIZE
4.71*x1 + 2.39*x2 + 2.77*x3 + 2.39*x4 + 1.99*x5 + 5.35*x6 + 3.44*x7 + 5.49*x8 +
0.0
SUBJECT TO
_C1: 0.5 x1 + 0.042 x2 + 0.084 x3 + 0.055 x4 + 0.058 x5 + 0.1 x6 + 0.084 x7
 + 0.084 x8 <= 3450

_C2: 0.3 x1 + 0.025 x2 + 0.05 x3 + 0.03 x4 + 0.032 x5 + 0.06 x6 + 0.05 x7
 + 0.05 x8 <= 10000

_C3: 0.01 x1 + 0.0008 x2 + 0.0016 x3 + 0.0008 x4 + 0.0008 x5 + 0.003 x6
 + 0.0016 x7 + 0.0016 x8 <= 70

_C4: 0.001 x4 + 0.0008 x5 <= 5

_C5: 0.01 x1 + 0.0008 x2 + 0.0016 x3 + 0.001 x4 + 0.001 x5 + 0.002 x6
 + 0.0016 x7 + 0.0016 x8 <= 70

_C6: 0.00025 x2 + 0.0005 x3 + 0.008 x4 + 0.007 x5 + 0.0007 x6 + 0.0005 x7
 + 0.0005 x8 <= 20

_C7: 0.00025 x2 + 0.0005 x3 + 0.0006 x4 + 0.0008 x5 + 0.0005 x6 + 0.0005 x7
 + 0.0005 x8 <= 6

_C8: 0.25 x4 + 0.25 x5 <= 400

_C9: 0.03 x4 + 0.04 x5 <= 55

_C10: 0.04 x6 + 0.025 x7 <= 100

_C11: 0.04 x6 + 0.025 x7 + 0.025 x8 <= 120

_C12: 0.025 x6 <= 40

_C13: 0.0002 x6 <= 0.5

_C14: 0.04 x8 <= 40

_C15: 0.015 x8 <= 15

VARIABLES
```

```
x1 Continuous
x2 Continuous
x3 Continuous
x4 Continuous
x5 Continuous
x6 Continuous
x7 Continuous
x8 Continuous
```

[15]:
```
status = Lp_prob.solve()
print(p.LpStatus[status])
```

```
Optimal
```

[18]:
```
print(p.value(x1), p.value(x2), p.value(x3), p.value(x4), p.value(x5), p.
↪value(x6), p.value(x7), p.value(x8), p.value(Lp_prob.objective))
```

```
4832.8 18800.0 0.0 0.0 0.0 1600.0 0.0 1000.0 81744.488
```

[ ]:
```

```