Modeling libries in python

A brief introduction to modeling libraries in Python

Kunal Khurana

2024-03-01

Table of contents

Interfacing between pandas and model code

Creating model descriptions with patsy

- Data Transformations in Pasty Formulas
- Categorical data and patsy

Introduction to statsmodels

- Estimating linear models
- Estimating time series processes

Introduction to scitkit-learn

Interface (data loading and cleaning beforing model building)

```
import pandas as pd
import numpy as np

import patsy
import statsmodels
```

```
data = pd.DataFrame({
    "x" :[1, 2, 3, 4, 5],
    "y" :[0.1, .2, 0.4, .6, .7],
    "z" :[-1, -3, -.45, -5.6, 4]
})
```

```
data
```

```
data.to_numpy()
```

```
# working with DataFrames
df2 = pd.DataFrame(data.to_numpy(),
                  columns= ['one', 'two', 'three'])
```

```
df2
```

```
df3 = data.copy()
```

```
df3['strings'] = ['a', 'b', 'c', 'd', 'e']

df3
```

```
df3.to_numpy()
```

```
# to use subset of columns, loc indexing with to_numpy

model_cols = ["x", "y"]
```

```
data.loc[:, model_cols].to_numpy()
```

```
data['category'] =pd.Categorical(['a', 'b', 'a','a', 'b'],
                                  categories = ['a', 'b'])

data
```

```
# replacing category with dummy variables
dummies = pd.get_dummies(data.category,  prefix='category')

data_with_dummies = data.drop('category', axis=1).join(dummies)

data_with_dummies
```

Creating model descriptions with Patsy

```
data
```

```
data.drop('category', axis=1)
```

```
a, b = patsy.dmatrices('z ~ x + y', data)
```

```
a
```

```
DesignMatrix with shape (5, 1)
      z
  -1.00
  -3.00
  -0.45
  -5.60
   4.00
  Terms:
    'z' (column 0)
```

```
b
```

```
DesignMatrix with shape (5, 3)
```

```
  Intercept  x    y
          1  1  0.1
          1  2  0.2
          1  3  0.4
          1  4  0.6
          1  5  0.7
  Terms:
    'Intercept' (column 0)
    'x' (column 1)
    'y' (column 2)
```

```
np.asarray(a)
```

```
np.asarray(b)
```

```
# adding +o to suppress the intercept
patsy.dmatrices('z ~ x + y + 0', data)[1]
```

```
patsy.dmatrices('z ~ x + y + 0', data)
```

passing pasty objects directly into algorithms

-eg. numpy.linalg.lstsq

```
rcond = -1
coef, resid, _, _ = np.linalg.lstsq(a, b)
```

```
C:\Users\Khurana_Kunal\AppData\Local\Temp\ipykernel_23924\735709127.py:2: FutureWarning: `rc
To use the future default and silence this warning we advise to pass `rcond=None`, to keep u
  coef, resid, _, _ = np.linalg.lstsq(a, b)
```

```
coef
```

```
array([[-0.10510315, -0.18675353, -0.02501629]])
```

```
coef = pd.Series(coef.squeeze(), index=a.design_info.column_names)
```

Data Transformations in Pastry Formulas

```python
a, b = patsy.dmatrices('x ~ y + np.log(np.abs(x) + 1)', data)
```

```python
b
```

```
DesignMatrix with shape (5, 3)
  Intercept    y  np.log(np.abs(x) + 1)
          1  0.1                0.69315
          1  0.2                1.09861
          1  0.4                1.38629
          1  0.6                1.60944
          1  0.7                1.79176
  Terms:
    'Intercept' (column 0)
    'y' (column 1)
    'np.log(np.abs(x) + 1)' (column 2)
```

```python
a,b = patsy.dmatrices("y ~ standardize(x) + center(z)", data)
```

```python
a
```

```
DesignMatrix with shape (5, 1)
    y
  0.1
  0.2
  0.4
  0.6
  0.7
  Terms:
    'y' (column 0)
```

```python
b
```

```
DesignMatrix with shape (5, 3)
  Intercept  standardize(x)  center(z)
          1        -1.41421       0.21
          1        -0.70711      -1.79
```

```
         1          0.00000       0.76
         1          0.70711      -4.39
         1          1.41421       5.21
  Terms:
    'Intercept' (column 0)
    'standardize(x)' (column 1)
    'center(z)' (column 2)
```

Categorical Data and Patsy

```python
data2 = pd.DataFrame({
    'key1' : ['a', 'b', 'c', 'a', 'c', 'b'],
    'key2' : [0, 1, 0, 1, 0, 1],
    'v1' : [1, 2, 3, 4, 5, 6],
    'v2' : [-1, 0, -1.5, 4.0, 2.5, -1.7]
})
```

```python
y, X = patsy.dmatrices('v2 ~ key1', data2)
```

```python
X
```

```
DesignMatrix with shape (6, 3)
  Intercept  key1[T.b]  key1[T.c]
          1          0          0
          1          1          0
          1          0          1
          1          0          0
          1          0          1
          1          1          0
  Terms:
    'Intercept' (column 0)
    'key1' (columns 1:3)
```

```python
y, X = patsy.dmatrices('v2 ~ C(key2)', data2)
```

```python
X
```

```
DesignMatrix with shape (6, 2)
```

```
  Intercept  C(key2)[T.1]
          1             0
          1             1
          1             0
          1             1
          1             0
          1             1
  Terms:
    'Intercept' (column 0)
    'C(key2)' (column 1)
```

```
data2['key2'] = data2['key2'].map({0: 'zero', 1:'one'})
```

```
data2
```

	key1	key2	v1	v2
0	a	zero	1	-1.0
1	b	one	2	0.0
2	c	zero	3	-1.5
3	a	one	4	4.0
4	c	zero	5	2.5
5	b	one	6	-1.7

```
y, X = patsy.dmatrices('v2 ~ key1 + key2', data2)
```

```
X
```

```
DesignMatrix with shape (6, 4)
  Intercept  key1[T.b]  key1[T.c]  key2[T.zero]
          1          0          0             1
          1          1          0             0
          1          0          1             1
          1          0          0             0
          1          0          1             1
          1          1          0             0
  Terms:
    'Intercept' (column 0)
    'key1' (columns 1:3)
    'key2' (column 3)
```

```python
y, X = patsy.dmatrices('v2 ~ key1 + key2 + key1:key2', data2)
```

```python
X
```

```
DesignMatrix with shape (6, 6)
  Columns:
    ['Intercept',
     'key1[T.b]',
     'key1[T.c]',
     'key2[T.zero]',
     'key1[T.b]:key2[T.zero]',
     'key1[T.c]:key2[T.zero]']
  Terms:
    'Intercept' (column 0)
    'key1' (columns 1:3)
    'key2' (column 3)
    'key1:key2' (columns 4:6)
  (to view full data, use np.asarray(this_obj))
```

Statsmodles

- linear models, generalized linear models, and robust linear models
- linear mixed effects models
- ANOVA
- time series and state space models
- generalized methods of moments

Estimating linear models

```python
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

```python
# generating a linear model from a random data
rng = np.random.default_rng(seed = 12345)

def dnorm(mean, variance, size=1):
    if isinstance (size, int):
        size = size,
```

```
    return mean+ np.sqrt(variance) * rng.standard_normal(*size)
```

```
N = 100
X = np.c_[dnorm(0, 0.4, size=N),
         dnorm(0, 0.6, size=N),
         dnorm(0, 0.2, size= N)]
eps = dnorm(0, 0.1, size=N)
beta = [0.1, 0.3, 0.5]

y = np.dot(X, beta) + eps
```

```
X[:5]
```

```
array([[-0.90050602, -0.18942958, -1.0278702 ],
       [ 0.79925205, -1.54598388, -0.32739708],
       [-0.55065483, -0.12025429,  0.32935899],
       [-0.16391555,  0.82403985,  0.20827485],
       [-0.04765129, -0.21314698, -0.04824364]])
```

```
y[:5]
```

```
array([-0.59952668, -0.58845445,  0.18563386, -0.00747657, -0.01537445])
```

```
# fitting a linear model with intercept term
X_model = sm.add_constant(X)

X_model[:5]
```

```
array([[ 1.        , -0.90050602, -0.18942958, -1.0278702 ],
       [ 1.        ,  0.79925205, -1.54598388, -0.32739708],
       [ 1.        , -0.55065483, -0.12025429,  0.32935899],
       [ 1.        , -0.16391555,  0.82403985,  0.20827485],
       [ 1.        , -0.04765129, -0.21314698, -0.04824364]])
```

```
# filling least squre linear regression with sm.OLS
model = sm.OLS(y, X)
```

```python
results = model.fit()
```

```python
results.params
```

```
array([0.06681503, 0.26803235, 0.45052319])
```

```python
# printing summary
print(results.summary())
```

```
                            OLS Regression Results
=======================================================================================
Dep. Variable:                      y   R-squared (uncentered):                   0.469
Model:                            OLS   Adj. R-squared (uncentered):              0.452
Method:                 Least Squares   F-statistic:                              28.51
Date:                Fri, 01 Mar 2024   Prob (F-statistic):                    2.66e-13
Time:                        12:55:52   Log-Likelihood:                         -25.611
No. Observations:                 100   AIC:                                      57.22
Df Residuals:                      97   BIC:                                      65.04
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1             0.0668      0.054      1.243      0.217      -0.040       0.174
x2             0.2680      0.042      6.313      0.000       0.184       0.352
x3             0.4505      0.068      6.605      0.000       0.315       0.586
==============================================================================
Omnibus:                        0.435   Durbin-Watson:                   1.869
Prob(Omnibus):                  0.805   Jarque-Bera (JB):                0.301
Skew:                           0.134   Prob(JB):                        0.860
Kurtosis:                       2.995   Cond. No.                         1.64
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

```python
data = pd.DataFrame(X, columns=['col0', 'col1', 'col2'])
```

```python
data['y']=y
data[:5]
```

	col0	col1	col2	y
0	-0.900506	-0.189430	-1.027870	-0.599527
1	0.799252	-1.545984	-0.327397	-0.588454
2	-0.550655	-0.120254	0.329359	0.185634
3	-0.163916	0.824040	0.208275	-0.007477
4	-0.047651	-0.213147	-0.048244	-0.015374

```python
# using statsmodles formula API and Pastry formuls
results = smf.ols('y ~ col0 + col1 + col2', data=data).fit()
```

```python
results.params
```

```
Intercept   -0.020799
col0         0.065813
col1         0.268970
col2         0.449419
dtype: float64
```

```python
results.tvalues
```

```
Intercept   -0.652501
col0         1.219768
col1         6.312369
col2         6.567428
dtype: float64
```

```python
# computing predicted values
results.predict(data[:5])
```

```
0   -0.592959
1   -0.531160
2    0.058636
3    0.283658
4   -0.102947
dtype: float64
```

Estimating time series processes

```
init_x = 4
values = [init_x, init_x]
N = 1000
b0 = 0.8
b1 = -0.4
noise = dnorm(0, 0.1, N)
for i in range(N):
    new_x = values[-1] * b0 + values[-2] +b1 + noise[i]
    values.append(new_x)
```

```
from statsmodels.tsa.ar_model import AutoReg

MAXLAGS = 5

model = AutoReg(values, MAXLAGS)

results = model.fit()
```

```
C:\Users\Khurana_Kunal\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8
  cov_p = self.normalized_cov_params * scale
```

```
results.params
```

```
array([0.        , 0.81652213, 0.5528124 , 0.37427221, 0.25339463,
       0.17155651])
```

Scikit-learn

```
! pip install scikit-learn
```

```
train = pd.read_csv(r"E:\pythonfordatanalysis\semainedu26fevrier\train (1).csv")
```

```
test= pd.read_csv(r"E:\pythonfordatanalysis\semainedu26fevrier\test (1).csv")
```

```
train.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	Sib
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0

```
# looking for missing data
train.isna().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
test.isna().sum()
```

```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
```

```
Embarked         0
dtype: int64
```

```
# using 'Age' as a predictor
# using median of training set to fill missing values

impute_value = train['Age'].median()

train['Age'] = train['Age'].fillna(impute_value)

test['Age'] = test['Age'].fillna(impute_value)
```

```
# specying the models
train['isFemale'] = (train['Sex'] == 'female').astype(int)

test['isFemale'] = (test['Sex'] == 'female').astype(int)
```

```
# creating NumPy arrays and deciding on some model variables
predictors = ['Pclass', 'isFemale', 'Age']

X_train = train[predictors].to_numpy()

X_test = test[predictors].to_numpy()

y_train = train['Survived'].to_numpy()

X_train[:5]
```

```
array([[ 3.,  0., 22.],
       [ 1.,  1., 38.],
       [ 3.,  1., 26.],
       [ 1.,  1., 35.],
       [ 3.,  0., 35.]])
```

```
y_train[:5]
```

```
array([0, 1, 1, 1, 0], dtype=int64)
```

```python
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
```

```python
model.fit(X_train, y_train)
LogisticRegression()
```

```
LogisticRegression()
```

```python
y_predict = model.predict(X_test)
```

```python
y_predict[:10]
```

```
array([0, 0, 0, 0, 1, 0, 1, 0, 1, 0], dtype=int64)
```

```python
(y_true == y_predict).mean()
```

Theory

- many models have parameters that can be tuned to avoid overfitting

  example- Cross-validataion (longer to train, but performs better)

```python
from sklearn.linear_model import LogisticRegressionCV
```

```python
model_cv = LogisticRegressionCV(Cs=10)

model_cv.fit(X_train, y_train)

LogisticRegressionCV()
```

```
LogisticRegressionCV()
```

```python
from sklearn.model_selection import cross_val_score
```

```python
model = LogisticRegression (C=10)

scores = cross_val_score(model, X_train, y_train, cv=4)

scores
```

```
array([0.77578475, 0.79820628, 0.77578475, 0.78828829])
```