Introduction to Modeling Libraries

In [1]:
```
import numpy as np
import pandas as pd
np.random.seed(12345)
import matplotlib.pyplot as plt
plt.rc('figure', figsize=(10, 6))
PREVIOUS_MAX_ROWS = pd.options.display.max_rows
pd.options.display.max_rows = 20
np.set_printoptions(precision=4, suppress=True)
```

Interfacing Between pandas and Model Code

In [2]:
```
import pandas as pd
import numpy as np
data = pd.DataFrame({
    'x0': [1, 2, 3, 4, 5],
    'x1': [0.01, -0.01, 0.25, -4.1, 0.],
    'y': [-1.5, 0., 3.6, 1.3, -2.]})
data
```

Out[2]:

	x0	x1	y
0	1	0.01	-1.5
1	2	-0.01	0.0
2	3	0.25	3.6
3	4	-4.10	1.3
4	5	0.00	-2.0

In [3]:
```
data.columns
```

Out[3]:
```
Index(['x0', 'x1', 'y'], dtype='object')
```

In [4]:
```
data.values
```

Out[4]:
```
array([[ 1.  ,  0.01, -1.5 ],
       [ 2.  , -0.01,  0.  ],
       [ 3.  ,  0.25,  3.6 ],
       [ 4.  , -4.1 ,  1.3 ],
       [ 5.  ,  0.  , -2.  ]])
```

In [5]:
```
df2 = pd.DataFrame(data.values, columns=['one', 'two', 'three'])
df2
```

Out[5]:

	one	two	three
0	1.0	0.01	-1.5
1	2.0	-0.01	0.0
2	3.0	0.25	3.6
3	4.0	-4.10	1.3
4	5.0	0.00	-2.0

In [6]:

```
df3 = data.copy()
df3['strings'] = ['a', 'b', 'c', 'd', 'e']
df3
```

Out[6]:

	x0	x1	y	strings
0	1	0.01	-1.5	a
1	2	-0.01	0.0	b
2	3	0.25	3.6	c
3	4	-4.10	1.3	d
4	5	0.00	-2.0	e

In [7]:

```
df3.values
```

Out[7]:

```
array([[1, 0.01, -1.5, 'a'],
       [2, -0.01, 0.0, 'b'],
       [3, 0.25, 3.6, 'c'],
       [4, -4.1, 1.3, 'd'],
       [5, 0.0, -2.0, 'e']], dtype=object)
```

In [8]:

```
model_cols = ['x0', 'x1']
data.loc[:, model_cols].values
```

Out[8]:

```
array([[ 1.  ,  0.01],
       [ 2.  , -0.01],
       [ 3.  ,  0.25],
       [ 4.  , -4.1 ],
       [ 5.  ,  0.  ]])
```

In [9]:

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

Out[9]:

	x0	x1	y	category
0	1	0.01	-1.5	a
1	2	-0.01	0.0	b
2	3	0.25	3.6	a
3	4	-4.10	1.3	a
4	5	0.00	-2.0	b

In [10]:

```
dummies = pd.get_dummies(data.category, prefix='category')
data_with_dummies = data.drop('category', axis=1).join(dummies)
data_with_dummies
```

Out[10]:

	x0	x1	y	category_a	category_b
0	1	0.01	-1.5	1	0
1	2	-0.01	0.0	0	1
2	3	0.25	3.6	1	0
3	4	-4.10	1.3	1	0
4	5	0.00	-2.0	0	1

Creating Model Descriptions with Patsy

y ~ x0 + x1

In [11]:
```
data = pd.DataFrame({
    'x0': [1, 2, 3, 4, 5],
    'x1': [0.01, -0.01, 0.25, -4.1, 0.],
    'y': [-1.5, 0., 3.6, 1.3, -2.]})
data
```

Out[11]:

	x0	x1	y
0	1	0.01	-1.5
1	2	-0.01	0.0
2	3	0.25	3.6
3	4	-4.10	1.3
4	5	0.00	-2.0

In [12]:
```
import patsy
y, X = patsy.dmatrices('y ~ x0 + x1', data)
```

In [13]:
```
y
```

Out[13]:
```
DesignMatrix with shape (5, 1)
     y
  -1.5
   0.0
   3.6
   1.3
  -2.0
  Terms:
    'y' (column 0)
```

In [14]:
```
X
```

Out[14]:
```
DesignMatrix with shape (5, 3)
  Intercept  x0     x1
          1   1   0.01
          1   2  -0.01
          1   3   0.25
          1   4  -4.10
          1   5   0.00
  Terms:
    'Intercept' (column 0)
    'x0' (column 1)
    'x1' (column 2)
```

In [15]:
```
np.asarray(y)
```

Out[15]:
```
array([[-1.5],
       [ 0. ],
       [ 3.6],
       [ 1.3],
       [-2. ]])
```

In [16]:
```
np.asarray(X)
```

Out[16]:
```
array([[ 1.  ,  1.  ,  0.01],
       [ 1.  ,  2.  , -0.01],
       [ 1.  ,  3.  ,  0.25],
       [ 1.  ,  4.  , -4.1 ],
       [ 1.  ,  5.  ,  0.  ]])
```

In [17]:
```
patsy.dmatrices('y ~ x0 + x1 + 0', data)[1]
```

Out[17]:
```
DesignMatrix with shape (5, 2)
  x0     x1
   1   0.01
   2  -0.01
   3   0.25
   4  -4.10
   5   0.00
  Terms:
    'x0' (column 0)
    'x1' (column 1)
```

In [18]:
```
coef, resid, _, _ = np.linalg.lstsq(X, y)
```

```
C:\Users\Usuário\AppData\Local\Temp\ipykernel_18184\2525922789.py:1: FutureWarnin
g: `rcond` parameter will change to the default of machine precision times ``max
(M, N)`` where M and N are the input matrix dimensions.
To use the future default and silence this warning we advise to pass `rcond=None`,
to keep using the old, explicitly pass `rcond=-1`.
  coef, resid, _, _ = np.linalg.lstsq(X, y)
```

In [19]:
```
coef
```

Out[19]:
```
array([[ 0.3129],
       [-0.0791],
       [-0.2655]])
```

In [20]:
```
coef = pd.Series(coef.squeeze(), index=X.design_info.column_names)
coef
```

Out[20]:
```
Intercept    0.312910
x0          -0.079106
x1          -0.265464
dtype: float64
```

Data Transformations in Patsy Formulas

In [21]:
```
y, X = patsy.dmatrices('y ~ x0 + np.log(np.abs(x1) + 1)', data)
X
```

Out[21]:
```
DesignMatrix with shape (5, 3)
  Intercept  x0  np.log(np.abs(x1) + 1)
          1   1                 0.00995
          1   2                 0.00995
          1   3                 0.22314
          1   4                 1.62924
          1   5                 0.00000
  Terms:
    'Intercept' (column 0)
    'x0' (column 1)
    'np.log(np.abs(x1) + 1)' (column 2)
```

In [22]:
```
y, X = patsy.dmatrices('y ~ standardize(x0) + center(x1)', data)
X
```

Out[22]:
```
DesignMatrix with shape (5, 3)
  Intercept  standardize(x0)  center(x1)
          1         -1.41421        0.78
          1         -0.70711        0.76
          1          0.00000        1.02
          1          0.70711       -3.33
          1          1.41421        0.77
  Terms:
    'Intercept' (column 0)
    'standardize(x0)' (column 1)
    'center(x1)' (column 2)
```

In [23]:
```
new_data = pd.DataFrame({
    'x0': [6, 7, 8, 9],
    'x1': [3.1, -0.5, 0, 2.3],
    'y': [1, 2, 3, 4]})
new_X = patsy.build_design_matrices([X.design_info], new_data)
new_X
```

Out[23]:
```
[DesignMatrix with shape (4, 3)
   Intercept  standardize(x0)  center(x1)
           1          2.12132        3.87
           1          2.82843        0.27
           1          3.53553        0.77
           1          4.24264        3.07
   Terms:
     'Intercept' (column 0)
     'standardize(x0)' (column 1)
     'center(x1)' (column 2)]
```

In [24]:
```
y, X = patsy.dmatrices('y ~ I(x0 + x1)', data)
X
```

Out[24]:
```
DesignMatrix with shape (5, 2)
  Intercept  I(x0 + x1)
          1        1.01
          1        1.99
          1        3.25
          1       -0.10
          1        5.00
  Terms:
    'Intercept' (column 0)
    'I(x0 + x1)' (column 1)
```

Categorical Data and Patsy

In [25]:
```
data = pd.DataFrame({
    'key1': ['a', 'a', 'b', 'b', 'a', 'b', 'a', 'b'],
    'key2': [0, 1, 0, 1, 0, 1, 0, 0],
```

```
    'v1': [1, 2, 3, 4, 5, 6, 7, 8],
    'v2': [-1, 0, 2.5, -0.5, 4.0, -1.2, 0.2, -1.7]
})
y, X = patsy.dmatrices('v2 ~ key1', data)
X
```

Out[25]:

```
DesignMatrix with shape (8, 2)
  Intercept  key1[T.b]
          1          0
          1          0
          1          1
          1          1
          1          0
          1          1
          1          0
          1          1
  Terms:
    'Intercept' (column 0)
    'key1' (column 1)
```

In [26]:

```
y, X = patsy.dmatrices('v2 ~ key1 + 0', data)
X
```

Out[26]:

```
DesignMatrix with shape (8, 2)
  key1[a]  key1[b]
        1        0
        1        0
        0        1
        0        1
        1        0
        0        1
        1        0
        0        1
  Terms:
    'key1' (columns 0:2)
```

In [27]:

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

Out[27]:

```
DesignMatrix with shape (8, 2)
  Intercept  C(key2)[T.1]
          1             0
          1             1
          1             0
          1             1
          1             0
          1             1
          1             0
          1             0
  Terms:
    'Intercept' (column 0)
    'C(key2)' (column 1)
```

In [28]:

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

Out[28]:

	key1	key2	v1	v2
0	a	zero	1	-1.0
1	a	one	2	0.0
2	b	zero	3	2.5
3	b	one	4	-0.5
4	a	zero	5	4.0
5	b	one	6	-1.2
6	a	zero	7	0.2
7	b	zero	8	-1.7

In [29]:

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

Out[29]:

```
DesignMatrix with shape (8, 3)
  Intercept  key1[T.b]  key2[T.zero]
          1          0             1
          1          0             0
          1          1             1
          1          1             0
          1          0             1
          1          1             0
          1          0             1
          1          1             1
  Terms:
    'Intercept' (column 0)
    'key1' (column 1)
    'key2' (column 2)
```

In [30]:

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

Out[30]:

```
DesignMatrix with shape (8, 4)
  Intercept  key1[T.b]  key2[T.zero]  key1[T.b]:key2[T.zero]
          1          0             1                       0
          1          0             0                       0
          1          1             1                       1
          1          1             0                       0
          1          0             1                       0
          1          1             0                       0
          1          0             1                       0
          1          1             1                       1
  Terms:
    'Intercept' (column 0)
    'key1' (column 1)
    'key2' (column 2)
    'key1:key2' (column 3)
```

Introduction to statsmodels

Estimating Linear Models

In [31]:

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

In [32]:
```
def dnorm(mean, variance, size=1):
    if isinstance(size, int):
        size = size,
    return mean + np.sqrt(variance) * np.random.randn(*size)

# For reproducibility
np.random.seed(12345)

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
```

In [33]:
```
X[:5]
```

Out[33]:
```
array([[-0.1295, -1.2128,  0.5042],
       [ 0.3029, -0.4357, -0.2542],
       [-0.3285, -0.0253,  0.1384],
       [-0.3515, -0.7196, -0.2582],
       [ 1.2433, -0.3738, -0.5226]])
```

In [34]:
```
y[:5]
```

Out[34]:
```
array([ 0.4279, -0.6735, -0.0909, -0.4895, -0.1289])
```

In [ ]:
```
X_model = sm.add_constant(X)
X_model[:5]
```

In [35]:
```
model = sm.OLS(y, X)
```

In [36]:
```
results = model.fit()
results.params
```

Out[36]:
```
array([0.1783, 0.223 , 0.501 ])
```

In [37]:
```
print(results.summary())
```

```
                                 OLS Regression Results
==================================================================================
=====
Dep. Variable:                      y   R-squared (uncentered):
0.430
Model:                            OLS   Adj. R-squared (uncentered):
0.413
Method:                 Least Squares   F-statistic:
24.42
Date:                Tue, 10 Sep 2024   Prob (F-statistic):                   7.4
4e-12
Time:                        22:40:14   Log-Likelihood:                        -3
4.305
No. Observations:                 100   AIC:
74.61
Df Residuals:                      97   BIC:
82.42
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1             0.1783      0.053      3.364      0.001       0.073       0.283
x2             0.2230      0.046      4.818      0.000       0.131       0.315
x3             0.5010      0.080      6.237      0.000       0.342       0.660
==============================================================================
Omnibus:                        4.662   Durbin-Watson:                   2.201
Prob(Omnibus):                  0.097   Jarque-Bera (JB):                4.098
Skew:                           0.481   Prob(JB):                        0.129
Kurtosis:                       3.243   Cond. No.                         1.74
==============================================================================

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 s
pecified.
```

In [38]:

```
data = pd.DataFrame(X, columns=['col0', 'col1', 'col2'])
data['y'] = y
data[:5]
```

Out[38]:

	col0	col1	col2	y
0	-0.129468	-1.212753	0.504225	0.427863
1	0.302910	-0.435742	-0.254180	-0.673480
2	-0.328522	-0.025302	0.138351	-0.090878
3	-0.351475	-0.719605	-0.258215	-0.489494
4	1.243269	-0.373799	-0.522629	-0.128941

In [39]:

```
results = smf.ols('y ~ col0 + col1 + col2', data=data).fit()
results.params
```

Out[39]:

```
Intercept    0.033559
col0         0.176149
col1         0.224826
col2         0.514808
dtype: float64
```

In [40]:

```
results.tvalues
```

Out[40]:
```
Intercept    0.952188
col0         3.319754
col1         4.850730
col2         6.303971
dtype: float64
```

In [41]:
```
results.predict(data[:5])
```

Out[41]:
```
0   -0.002327
1   -0.141904
2    0.041226
3   -0.323070
4   -0.100535
dtype: float64
```

Estimating Time Series Processes

In [42]:
```
init_x = 4

import random
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)
```

In [49]:
```
MAXLAGS = 5
model = sm.tsa.AutoReg(values, lags=MAXLAGS)
results = model.fit()
```

In [50]:
```
results.params
```

Out[50]:
```
array([-0.0062,  0.7845, -0.4085, -0.0136,  0.015 ,  0.0143])
```

Introduction to scikit-learn

In [51]:
```
train = pd.read_csv('datasets/titanic/train.csv')
test = pd.read_csv('datasets/titanic/test.csv')
train[:4]
```

Out[51]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123

In [52]:
```
test[:4]
```

Out[52]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S

In [53]:
```
train.isnull().sum()
```

Out[53]:
```
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
```

In [54]:
```
test.isnull().sum()
```

Out[54]:
```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

In [55]:
```
impute_value = train['Age'].median()
train['Age'] = train['Age'].fillna(impute_value)
test['Age'] = test['Age'].fillna(impute_value)
```

In [56]:
```
train.isnull().sum()
```

Out[56]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [57]:
```
test.isnull().sum()
```

Out[57]:
```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

In [58]:
```
train['IsFemale'] = (train['Sex'] == 'female').astype(int)
test['IsFemale'] = (test['Sex'] == 'female').astype(int)
```

In [59]:
```
predictors = ['Pclass', 'IsFemale', 'Age']
X_train = train[predictors].values
X_test = test[predictors].values
y_train = train['Survived'].values
X_train[:5]
```

Out[59]:
```
array([[ 3.,  0., 22.],
       [ 1.,  1., 38.],
       [ 3.,  1., 26.],
       [ 1.,  1., 35.],
       [ 3.,  0., 35.]])
```

In [60]:
```
y_train[:5]
```

Out[60]:
```
array([0, 1, 1, 1, 0], dtype=int64)
```

In [61]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [62]:
```
model.fit(X_train, y_train)
```

Out[62]:
```
LogisticRegression()
```

In [63]:
```
y_predict = model.predict(X_test)
y_predict[:10]
```

Out[63]:
```
array([0, 0, 0, 0, 1, 0, 1, 0, 1, 0], dtype=int64)
```

(y_true == y_predict).mean()

In [65]:
```
from sklearn.linear_model import LogisticRegressionCV
model_cv = LogisticRegressionCV(cv=10)
model_cv.fit(X_train, y_train)
```

Out[65]:
```
LogisticRegressionCV(cv=10)
```

In [66]:
```
from sklearn.model_selection import cross_val_score
model = LogisticRegression(C=10)
scores = cross_val_score(model, X_train, y_train, cv=4)
scores
```

Out[66]:
```
array([0.7758, 0.7982, 0.7758, 0.7883])
```

Continuing Your Education

In [ ]:
```
pd.options.display.max_rows = PREVIOUS_MAX_ROWS
```