Pre_processing

December 2, 2020

[1]:
```
import numpy as np
import pandas as pd

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression
```

[2]:
```
data = pd.read_csv('Data_full.csv')
```

[3]:
```
#display data
data
```

[3]:
```
       Administrative  Administrative_Duration  Informational  \
0                 0.0                      0.0            0.0
1                 0.0                      0.0            0.0
2                 0.0                     -1.0            0.0
3                 0.0                      0.0            0.0
4                 0.0                      0.0            0.0
…                …                       …             …
12325             3.0                    145.0            0.0
12326             0.0                      0.0            0.0
12327             0.0                      0.0            0.0
12328             4.0                     75.0            0.0
12329             0.0                      0.0            0.0

       Informational_Duration  ProductRelated  ProductRelated_Duration  \
0                         0.0             1.0                 0.000000
1                         0.0             2.0                64.000000
2                        -1.0             1.0                -1.000000
3                         0.0             2.0                 2.666667
4                         0.0            10.0               627.500000
…                       …              …                      …
12325                     0.0            53.0              1783.791667
12326                     0.0             5.0               465.750000
12327                     0.0             6.0               184.250000
12328                     0.0            15.0               346.000000
```

```
12329                    0.0             3.0                21.250000

       BounceRates  ExitRates  PageValues  SpecialDay Month  OperatingSystems  \
0         0.200000   0.200000    0.000000         0.0   Feb                 1
1         0.000000   0.100000    0.000000         0.0   Feb                 2
2         0.200000   0.200000    0.000000         0.0   Feb                 4
3         0.050000   0.140000    0.000000         0.0   Feb                 3
4         0.020000   0.050000    0.000000         0.0   Feb                 3
…              …          …           …         …   …                …
12325     0.007143   0.029031   12.241717         0.0   Dec                 4
12326     0.000000   0.021333    0.000000         0.0   Nov                 3
12327     0.083333   0.086667    0.000000         0.0   Nov                 3
12328     0.000000   0.021053    0.000000         0.0   Nov                 2
12329     0.000000   0.066667    0.000000         0.0   Nov                 3

       Browser  Region  TrafficType        VisitorType  Weekend  Revenue
0            1       1            1  Returning_Visitor    False    False
1            2       1            2  Returning_Visitor    False    False
2            1       9            3  Returning_Visitor    False    False
3            2       2            4  Returning_Visitor    False    False
4            3       1            4  Returning_Visitor     True    False
…          …     …          …                …      …      …
12325        6       1            1  Returning_Visitor     True    False
12326        2       1            8  Returning_Visitor     True    False
12327        2       1           13  Returning_Visitor     True    False
12328        2       3           11  Returning_Visitor    False    False
12329        2       1            2        New_Visitor     True    False

[12330 rows x 18 columns]
```

[4]:
```
#gather data types of the different data entries found in the file
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12330 entries, 0 to 12329
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Administrative           12316 non-null  float64
 1   Administrative_Duration  12316 non-null  float64
 2   Informational            12316 non-null  float64
 3   Informational_Duration   12316 non-null  float64
 4   ProductRelated           12316 non-null  float64
 5   ProductRelated_Duration  12316 non-null  float64
 6   BounceRates              12316 non-null  float64
 7   ExitRates                12316 non-null  float64
 8   PageValues               12330 non-null  float64
```

```
 9   SpecialDay               12330 non-null  float64
 10  Month                    12330 non-null  object
 11  OperatingSystems         12330 non-null  int64
 12  Browser                  12330 non-null  int64
 13  Region                   12330 non-null  int64
 14  TrafficType              12330 non-null  int64
 15  VisitorType              12330 non-null  object
 16  Weekend                  12330 non-null  bool
 17  Revenue                  12330 non-null  bool
dtypes: bool(2), float64(10), int64(4), object(2)
memory usage: 1.5+ MB
```

1 Pre-processing

[5]:
```
#look for the missing values in each column
data.isna().sum()
```

[5]:
```
Administrative             14
Administrative_Duration    14
Informational              14
Informational_Duration     14
ProductRelated             14
ProductRelated_Duration    14
BounceRates                14
ExitRates                  14
PageValues                  0
SpecialDay                  0
Month                       0
OperatingSystems            0
Browser                     0
Region                      0
TrafficType                 0
VisitorType                 0
Weekend                     0
Revenue                     0
dtype: int64
```

[6]:
```
#display data corresponding to columns that are missing entries"
data[data.isna().sum(axis=1).astype(bool)]
```

[6]:
```
      Administrative  Administrative_Duration  Informational  \
1065             NaN                      NaN            NaN
1132             NaN                      NaN            NaN
1133             NaN                      NaN            NaN
1134             NaN                      NaN            NaN
1135             NaN                      NaN            NaN
```

1136	NaN	NaN	NaN
1473	NaN	NaN	NaN
1474	NaN	NaN	NaN
1475	NaN	NaN	NaN
1476	NaN	NaN	NaN
2037	NaN	NaN	NaN
2038	NaN	NaN	NaN
2039	NaN	NaN	NaN
2753	NaN	NaN	NaN

	Informational_Duration	ProductRelated	ProductRelated_Duration \
1065	NaN	NaN	NaN
1132	NaN	NaN	NaN
1133	NaN	NaN	NaN
1134	NaN	NaN	NaN
1135	NaN	NaN	NaN
1136	NaN	NaN	NaN
1473	NaN	NaN	NaN
1474	NaN	NaN	NaN
1475	NaN	NaN	NaN
1476	NaN	NaN	NaN
2037	NaN	NaN	NaN
2038	NaN	NaN	NaN
2039	NaN	NaN	NaN
2753	NaN	NaN	NaN

	BounceRates	ExitRates	PageValues	SpecialDay	Month	OperatingSystems \
1065	NaN	NaN	0.0	0.0	Mar	2
1132	NaN	NaN	0.0	0.0	Mar	1
1133	NaN	NaN	0.0	0.0	Mar	2
1134	NaN	NaN	0.0	0.0	Mar	2
1135	NaN	NaN	0.0	0.0	Mar	3
1136	NaN	NaN	0.0	0.0	Mar	2
1473	NaN	NaN	0.0	0.0	Mar	2
1474	NaN	NaN	0.0	0.0	Mar	1
1475	NaN	NaN	0.0	0.0	Mar	2
1476	NaN	NaN	0.0	0.0	Mar	1
2037	NaN	NaN	0.0	0.0	Mar	3
2038	NaN	NaN	0.0	0.0	Mar	2
2039	NaN	NaN	0.0	0.0	Mar	3
2753	NaN	NaN	0.0	0.0	May	2

	Browser	Region	TrafficType	VisitorType	Weekend	Revenue
1065	2	2	1	Returning_Visitor	False	False
1132	1	1	2	Returning_Visitor	False	False
1133	4	5	1	Returning_Visitor	False	False
1134	2	1	2	Returning_Visitor	False	False

```
1135          2       1              1  Returning_Visitor    False     False
1136          2       1              2  Returning_Visitor    False     False
1473          2       1              1  Returning_Visitor     True     False
1474          1       6              1  Returning_Visitor     True     False
1475          2       3              1  Returning_Visitor    False     False
1476          1       2              3  Returning_Visitor    False     False
2037          2       4              1  Returning_Visitor    False     False
2038          2       1              2  Returning_Visitor    False     False
2039          2       4             15  Returning_Visitor     True     False
2753          2       4             13  Returning_Visitor    False     False
```

[7]:
```
# address missing data entries
data = data.dropna(axis=0).reset_index(drop=True)
```

2 Changing string entries to numeric

[8]:
```
# verify
print("Total missing values:", data.isna().sum().sum())

# print corrected data withou missing entries
# data
```

```
Total missing values: 0
```

[9]:
```
{column: list(data[column].unique()) for column in data.columns if data.
 ↪dtypes[column] == 'object'}
```

[9]:
```
{'Month': ['Feb',
  'Mar',
  'May',
  'Oct',
  'June',
  'Jul',
  'Aug',
  'Nov',
  'Sep',
  'Dec'],
 'VisitorType': ['Returning_Visitor', 'New_Visitor', 'Other']}
```

[10]:
```
def ordinal_encode(df, column, ordering):
    df = df.copy()
    df[column] = df[column].apply(lambda x: ordering.index(x))
    return df

def onehot_encode(df, column, prefix):
```

```python
    df = df.copy()
    dummies = pd.get_dummies(df[column], prefix=prefix)
    df = pd.concat([df, dummies], axis=1)
    df = df.drop(column, axis=1)
    return df
```

[11]:
```python
month_ordering = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'June', 'Jul', 'Aug',␣
 ↪'Sep', 'Oct', 'Nov', 'Dec']
visitor_prefix = 'V'
# encode data
data = ordinal_encode(data,'Month',month_ordering)
data = onehot_encode(data,'VisitorType',visitor_prefix)
data['Weekend'] = data['Weekend'].astype(np.int)
data['Revenue'] = data['Revenue'].astype(np.int)
```

[12]:
```python
# display encoded data
data
```

[12]:
```
       Administrative  Administrative_Duration  Informational  \
0                 0.0                      0.0            0.0
1                 0.0                      0.0            0.0
2                 0.0                     -1.0            0.0
3                 0.0                      0.0            0.0
4                 0.0                      0.0            0.0
…                 …                        …              …
12311             3.0                    145.0            0.0
12312             0.0                      0.0            0.0
12313             0.0                      0.0            0.0
12314             4.0                     75.0            0.0
12315             0.0                      0.0            0.0

       Informational_Duration  ProductRelated  ProductRelated_Duration  \
0                         0.0             1.0                 0.000000
1                         0.0             2.0                64.000000
2                        -1.0             1.0                -1.000000
3                         0.0             2.0                 2.666667
4                         0.0            10.0               627.500000
…                         …               …                        …
12311                     0.0            53.0              1783.791667
12312                     0.0             5.0               465.750000
12313                     0.0             6.0               184.250000
12314                     0.0            15.0               346.000000
12315                     0.0             3.0                21.250000

       BounceRates  ExitRates  PageValues  SpecialDay  Month  \
0         0.200000   0.200000    0.000000         0.0      1
1         0.000000   0.100000    0.000000         0.0      1
```

```
2        0.200000  0.200000   0.000000        0.0      1
3        0.050000  0.140000   0.000000        0.0      1
4        0.020000  0.050000   0.000000        0.0      1
…             …         …          …        …    …
12311    0.007143  0.029031  12.241717        0.0     11
12312    0.000000  0.021333   0.000000        0.0     10
12313    0.083333  0.086667   0.000000        0.0     10
12314    0.000000  0.021053   0.000000        0.0     10
12315    0.000000  0.066667   0.000000        0.0     10

       OperatingSystems  Browser  Region  TrafficType  Weekend  Revenue  \
0                     1        1       1            1        0        0
1                     2        2       1            2        0        0
2                     4        1       9            3        0        0
3                     3        2       2            4        0        0
4                     3        3       1            4        1        0
…                  …      …       …          …       …      …
12311                 4        6       1            1        1        0
12312                 3        2       1            8        1        0
12313                 3        2       1           13        1        0
12314                 2        2       3           11        0        0
12315                 3        2       1            2        1        0

       V_New_Visitor  V_Other  V_Returning_Visitor
0                  0        0                    1
1                  0        0                    1
2                  0        0                    1
3                  0        0                    1
4                  0        0                    1
…             …      …                   …
12311              0        0                    1
12312              0        0                    1
12313              0        0                    1
12314              0        0                    1
12315              1        0                    0

[12316 rows x 20 columns]
```

3 Splitting into training data and evaluation data

[13]:

```
y = data['Revenue'].copy()
X = data.drop('Revenue', axis=1)
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

```
X_train, X_eval, y_train, y_eval = train_test_split(X, y, train_size=0.8,␣
 ↪random_state=20)

print("Training data size Test Dataset")
print("Shape of X_train :", X_train.shape)
print("Shape of y_train :", y_train.shape)
print("Shape of X_eval :", X_eval.shape)
print("Shape of y_eval :", y_eval.shape)
```

```
Training data size Test Dataset
Shape of X_train : (9852, 19)
Shape of y_train : (9852,)
Shape of X_eval : (2464, 19)
Shape of y_eval : (2464,)
```

[14]:
```
import numpy as np
# from spicy.io import loadmat
import matplotlib.pyplot as plt

#A = np.genfromtxt('Data_Raw.csv', delimiter=',')
#print(A.dtype)

#Data = np.genfromtxt('Data_null.csv', delimiter=',')
#x_all = Data[0:12330,0:14] # features
#y_train = Data[0:12330,14] # corresponding labels

#x_train = Data[0:12330,0:14] # features
#y_train = Data[0:12330,14] # corresponding labels

# evaluation data
#x_eval= Data[1001:12330,0:14] # features
#y_eval = Data[1001:12330,14] # corresponding labels

# X = Data[0:3,0:14]
# y = Data[:,14]

# Classifier 1
#w = (X^T X)^(-1)X^T y
#X = x_train
#y = y_train
#w = np.linalg.inv(X.transpose()@X)@X.transpose()@y
#A = np.linalg.inv(X@X.T)

#print(np.round(w,2))
```

4 Training & evaluating - Least Squares

[15]:
```
# Classifier 1 - Training Data
#w = (X^T X)^(-1)X^T y
X = X_train
y = y_train
w_train = np.linalg.inv(X.transpose()@X)@X.transpose()@y
#A = np.linalg.inv(X@X.T)

print(np.round(w_train,2))
```

```
[ 0.    0.    0.01 -0.01  0.01  0.02  0.02 -0.05  0.17 -0.01  0.02 -0.01
  0.   -0.    0.    0.    0.   -0.01 -0.02]
```

[16]:
```
# all features
print('considering all features')
y_hat = np.sign(X_eval@w_train)

#error_vec = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat, y_test))]
#print('Errors: '+ str(sum(error_vec)))
#print('Percent error: '+str(100.0*sum(error_vec)/len(error_vec))+'%')
```

```
considering all features
```

[17]:
```
#print(np.round(y_hat,2))
#np.shape(y_hat)
#np.shape(y_eval)
```

[18]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np
import matplotlib.pyplot as plt

print('Performance of Least-Squares based classifier')
print('')
mse = mean_squared_error(y_eval, y_hat)
print('Mean squared error of testing set:', np.round(mse,4))
mae = mean_absolute_error(y_eval, y_hat)
print('Mean absolute error of testing set:', np.round(mae,4))
rmse = np.sqrt(mse)
print('Root Mean Squared Error of testing set:', np.round(rmse,4))
```

```
Performance of Least-Squares based classifier

Mean squared error of testing set: 0.9525
Mean absolute error of testing set: 0.9022
Root Mean Squared Error of testing set: 0.976
```

5 Training & evaluating - Truncated SVD

[19]:
```
import numpy as np
import scipy.io as sio

U, s, VT = np.linalg.svd(X_train,full_matrices=False)
#w = VT.T@np.diag(1/s)@U.T@y_train
#err_ = np.mean(np.sign(X_test@w) != y_test)
```

[20]:
```
#U, s, VT = np.linalg.svd(X_train)
np.shape(X_train)
```

[20]:
```
(9852, 19)
```

[21]:
```
U.shape, s.shape, VT.shape
```

[21]:
```
((9852, 19), (19,), (19, 19))
```

[22]:
```
#w = VT.T@np.diag(1/s)@U.T@y_train
w_svd = VT.T@np.diag(1/s)@U.T@y_train
```

[23]:
```
# all features
print('considering all features')
y_hat = np.sign(X_eval@w_svd)

#error_vec = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat, y_test))]
#print('Errors: '+ str(sum(error_vec)))
#print('Percent error: '+str(100.0*sum(error_vec)/len(error_vec))+'%')
```

```
considering all features
```

[24]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np
import matplotlib.pyplot as plt
#Calculating MSE, lower the value better it is. 0 means perfect prediction

print('Performance of Truncated SVD based classifier')
print('')

mse = mean_squared_error(y_eval, y_hat)
print('Mean squared error of testing set:', np.round(mse,4))
mae = mean_absolute_error(y_eval, y_hat)
print('Mean absolute error of testing set:', np.round(mae,4))
rmse = np.sqrt(mse)
print('Root Mean Squared Error of testing set:', np.round(rmse,4))
```

```
Performance of Truncated SVD based classifier
```

```
Mean squared error of testing set: 0.9412
Mean absolute error of testing set: 0.8965
Root Mean Squared Error of testing set: 0.9701
```

```
[ ]:
```