Final_Compiled

December 12, 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

data = pd.read_csv('Data_full.csv')

#look for the missing values in each column
data.isna().sum()

# address missing data entries
data = data.dropna(axis=0).reset_index(drop=True)

# verify
print("Total missing values:", data.isna().sum().sum())

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

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):
    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

month_ordering = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'June', 'Jul', 'Aug',␣
 ↪'Sep', 'Oct', 'Nov', 'Dec']
visitor_prefix = 'V'
```

```
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)

data
```

```
Total missing values: 0
```

[1]:

	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]
```

1 Splitting into training data and evaluation data

[2]:
```
y = data['Revenue'].copy()
X = data.drop('Revenue', axis=1)

scaler = StandardScaler()

X = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.3,␣
 ↪random_state=20)

print("Training dan Test Dataset")
print("Shape of X_train :", X_train.shape)
```

```
print("Shape of y_train :", y_train.shape)
print("Shape of X_test :", X_test.shape)
print("Shape of y_test :", y_test.shape)
```

```
Training dan Test Dataset
Shape of X_train : (3694, 19)
Shape of y_train : (3694,)
Shape of X_test : (8622, 19)
Shape of y_test : (8622,)
```

[3]:
```
#convert dataset from pandas frame to numpy dataset
y_train = y_train.values
```

2 Training & evaluating - First Classifier, Least Squares

[4]:
```
# 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.    0.01  0.01 -0.02 -0.    0.03  0.03 -0.06  0.16 -0.01  0.02 -0.02
 -0.01 -0.    0.    0.01 -0.    0.   -0.02]
```

[5]:
```
# all features
y_hat = np.sign(X_test@w_train)
print('considering all features', y_hat)
```

```
considering all features [-1. -1. -1. … -1. -1.  1.]
```

[6]:
```
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_test, y_hat)
print('Mean squared error of testing set:', np.round(mse,4))
mae = mean_absolute_error(y_test, 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: 0.9395
Mean absolute error of testing set: 0.8928
Root Mean Squared Error of testing set: 0.9693
```

3 Training & evaluating - Second Classifier - Truncated SVD

[7]:
```
min_err, min_r, min_w =np.inf,-1,None
err_sum = 0
for r in range(1,20):
    U, s, VT=np.linalg.svd(X_train)
    w = VT[:r, :].T@np.diag(1/s[:r])@U[:,:r].T@y_train
    err_ = np.mean(np.sign(X_test@w) != y_test)
    if err_<min_err:
        min_err, min_r, min_w = err_, r, w

    err_sum+=np.mean(np.sign(X_train@min_w)!=y_train)
```

[8]:
```
# all features
y_hat = np.sign(X_test@min_w)
print('considering all features', y_hat)
```

```
considering all features [-1. -1. -1. … -1. -1.  1.]
```

[9]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np
import matplotlib.pyplot as plt

print('Performance of Truncated SVD based classifier')
print('')
mse = mean_squared_error(y_test, y_hat)
print('Mean squared error of testing set:', np.round(mse,4))
mae = mean_absolute_error(y_test, 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: 0.9139
Mean absolute error of testing set: 0.8801
Root Mean Squared Error of testing set: 0.956
```

4 Training & evaluating - Third Classifier - Neural Networks (see independent code)

5 Optimization - Cross - Validation

[10]:
```
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

data = pd.read_csv('Data_full.csv')

#look for the missing values in each column
data.isna().sum()

# address missing data entries
data = data.dropna(axis=0).reset_index(drop=True)

# verify
print("Total missing values:", data.isna().sum().sum())

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

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):
    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

month_ordering = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'June', 'Jul', 'Aug',␣
 ↪'Sep', 'Oct', 'Nov', 'Dec']
visitor_prefix = 'V'


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)

data
```

```
Total missing values: 0
```

[10]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues	SpecialDay	Month
0	0.0	0.0	0.0	0.0	1.0	0.000000	0.200000	0.200000	0.000000	0.0	1
1	0.0	0.0	0.0	0.0	2.0	64.000000	0.000000	0.100000	0.000000	0.0	1
2	0.0	-1.0	0.0	-1.0	1.0	-1.000000	0.200000	0.200000	0.000000	0.0	1
3	0.0	0.0	0.0	0.0	2.0	2.666667	0.050000	0.140000	0.000000	0.0	1
4	0.0	0.0	0.0	0.0	10.0	627.500000	0.020000	0.050000	0.000000	0.0	1
…	…	…	…	…	…	…	…	…	…	…	…
12311	3.0	145.0	0.0	0.0	53.0	1783.791667	0.007143	0.029031	12.241717	0.0	11
12312	0.0	0.0	0.0	0.0	5.0	465.750000	0.000000	0.021333	0.000000	0.0	10
12313	0.0	0.0	0.0	0.0	6.0	184.250000	0.083333	0.086667	0.000000	0.0	10
12314	4.0	75.0	0.0	0.0	15.0	346.000000	0.000000	0.021053	0.000000	0.0	10
12315	0.0	0.0	0.0	0.0	3.0	21.250000	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]
```

[11]:
```
import numpy as np
import scipy.io as sio
y = data['Revenue'].copy()
X = data.drop('Revenue', axis=1)

scaler = StandardScaler()

X = scaler.fit_transform(X)
```

[12]:
```
#convert dataset from pandas frame to numpy dataset
h = y.values
```

[13]:
```
y = h
```

[14]:
```
y = data['Revenue'].copy()
X = data.drop('Revenue', axis=1)

scaler = StandardScaler()
```

```
X = scaler.fit_transform(X)

#X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7,␣
 ↪random_state=20)
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.3,␣
 ↪random_state=20)

print("Training dan Test Dataset")
# from sklearn.model_selection import train_test_split
# splitting the X, and y
# X_train, X_test, y_train, y_test =  train_test_split(X,y, test_size = 0.2,␣
 ↪random_state = 0)
# checking the shapes
print("Shape of X_train :", X_train.shape)
print("Shape of y_train :", y_train.shape)
print("Shape of X_test :", X_test.shape)
print("Shape of y_test :", y_test.shape)
```

```
Training dan Test Dataset
Shape of X_train : (3694, 19)
Shape of y_train : (3694,)
Shape of X_test : (8622, 19)
Shape of y_test : (8622,)
```

[15]:
```
err_sum = 0
for i in range(4):
    for j in range(4):
        if i == j: continue
        test_idx_1 = np.arange(i*3079, (i+1)*3079)
        test_idx_2 =np.arange(j*3079, (j+1)*3079)
        train_idx =np.setdiff1d(np.arange(12316), test_idx_1)
        train_idx =np.setdiff1d(train_idx, test_idx_2)
        X_train, y_train = X[train_idx, :], y[train_idx]
        X_test_1, y_test_1 =X[test_idx_1, :], y[test_idx_1]
        X_test_2, y_test_2 =X[test_idx_2, :], y[test_idx_2]
        min_err, min_r, min_w =np.inf,-1,None
        for r in range(1,20):
            U, s, VT=np.linalg.svd(X_train)
            w = VT[:r, :].T@np.diag(1/s[:r])@U[:,:r].T@y_train
            err_ = np.mean(np.sign(X_test_1@w) != y_test_1)
            if err_<min_err:
                min_err, min_r, min_w = err_, r, w

            err_sum+=np.mean(np.sign(X_test_2@min_w)!=y_test_2)

print(err_sum/4/3)
```

```
16.620358341452857
```