Neural_Networks_Final

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'
```

```python
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=0.7,␣
 ↪random_state=20)
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.15,␣
 ↪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 : (1847, 19)
Shape of y_train : (1847,)
Shape of X_test : (10469, 19)
Shape of y_test : (10469,)
```

[3]:

```
#convert dataset from pandas frame to numpy dataset
y_train = y_train.values
#y_train
```

2 Neural Networks

[4]:

```
import numpy as np
import matplotlib.pyplot as plt

X=X_train
Y=y_train

p = int(19) #features
n = len(y_train)
q = len(y_train) #number of classification problems

W = np.random.randn(p+1, q);
```

[5]:

```
## Train NN
Xb = np.hstack((np.ones((n,1)), X))
#q = np.shape(Y) #number of classification problems
M = 3 #number of hidden nodes

## initial weights
#W = np.random.randn(p+1, q);

alpha = 0.1 #step size
L = 10 #number of epochs
```

```python

def logsig(_x):
    return 1/(1+np.exp(-_x))

for epoch in range(L):
    ind = np.random.permutation(n)
    for i in ind:
        # Forward-propagate
        Yhat = logsig(Xb[[i],:]@W)
         # Backpropagate
        #delta = (Yhat-Y[[i],:])*Yhat*(1-Yhat)
        delta = (Yhat-Y[[i]])*Yhat*(1-Yhat)
        Wnew = W - alpha*Xb[[i],:].T@delta
        W = Wnew
    print(epoch)
```

```
0
1
2
3
4
5
6
7
8
9
```

[6]:

```python
Yhat=Yhat.T
```

[7]:

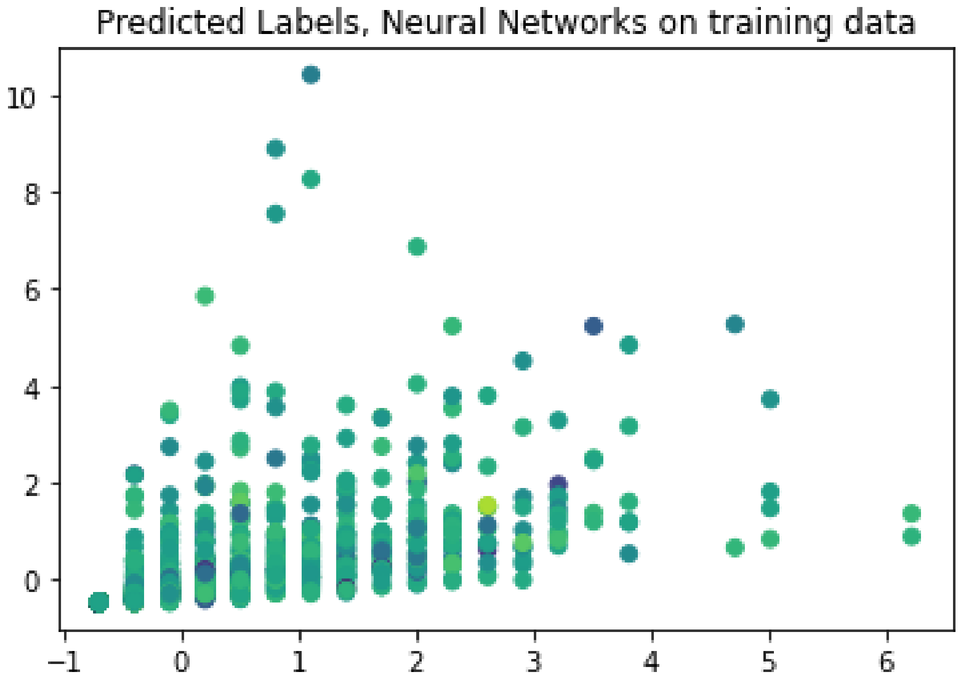
```python
np.shape(Yhat)
```

[7]:

```
(1847, 1)
```

[8]:

```python
plt.scatter(X[:,0], X[:,1], c=Yhat[:,0])
plt.title('Predicted Labels, Neural Networks on training data')
plt.show()
```



[9]:
```
err_c1 = np.sum(abs(np.round(Yhat[len(Yhat)-1])-Y))
print('Errors, neural network classifier:', err_c1)
```

```
Errors, neural network classifier: 280.0
```

[10]:
```
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
#mse = mean_squared_error(np.round(Yhat[len(Yhat)-1]), Y)
mse = mean_squared_error(np.round(Yhat), Y)
print('Mean Squared Error, neural network classifier on training data:',np.
 ↪round(mse*100,2))
```

```
Mean Squared Error, neural network classifier on training data: 15.16
```

[11]:
```
# validation test data
n2=len(y_test)
Xv = np.hstack((np.ones((n2,1)), X_test))
```

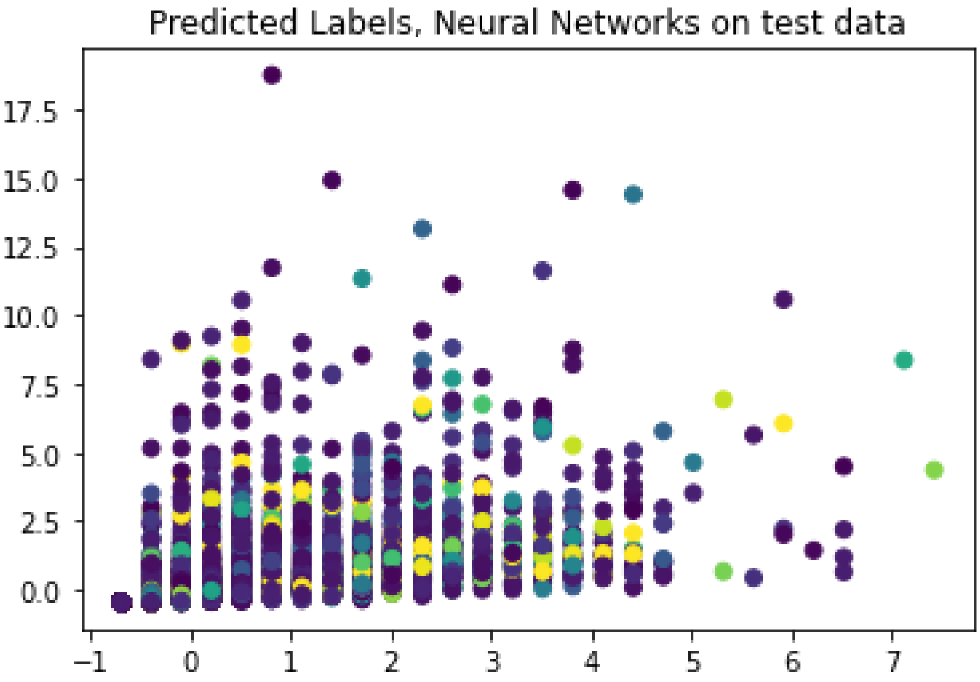
[12]:
```
An = np.hstack((np.ones((n2,1)), Xv@W))
```

[13]:
```
Hn = logsig(An)
```

[14]:
```
Yhatn = logsig(Xv@W)
```

[15]:

```
plt.scatter(X_test[:,0], X_test[:,1], c=Yhatn[:,0])
plt.title('Predicted Labels, Neural Networks on test data')
plt.show()
```



[16]:

```
err_c1 = np.sum(abs(np.round(Yhatn[len(Yhatn)-1])-y_train))
print('Errors, neural network classifier:', err_c1)
```

```
Errors, neural network classifier: 280.0
```

[17]:

```
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
#mse = mean_squared_error(np.round(Yhat[len(Yhat)-1]), Y)
mse = mean_squared_error(np.round(Yhat), Y)
print('Mean Squared Error, neural network classifier on training data:',np.
 ↪round(mse*100,2))
```

```
Mean Squared Error, neural network classifier on training data: 15.16
```

[ ]: