lab-10-data-mining

April 29, 2024

0.1 Classification using a Naive Bayes Classifier

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

df = pd.read_csv("/content/diabetes.csv")
```

[ ]:
```
df
```

[ ]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...  ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

[ ]:
```
y = df['Outcome']
y
```

[ ]:
```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

[ ]:
```
x = df.drop(columns = ['Outcome'])
def z_score(df):
    # copy the dataframe
    df_std = df.copy()
    # apply the z-score method
    for column in df_std.select_dtypes(include=np.number).columns:
        df_std[column] = (df_std[column] - df_std[column].mean()) /␣
 ↪df_std[column].std()

    return df_std

x = z_score(x)
x
```

[ ]:
```
     Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin       BMI  \
0       0.639530  0.847771       0.149543       0.906679 -0.692439  0.203880
1      -0.844335 -1.122665      -0.160441       0.530556 -0.692439 -0.683976
2       1.233077  1.942458      -0.263769      -1.287373 -0.692439 -1.102537
3      -0.844335 -0.997558      -0.160441       0.154433  0.123221 -0.493721
4      -1.141108  0.503727      -1.503707       0.906679  0.765337  1.408828
..           …       …              …              …       …       …
763     1.826623 -0.622237       0.356200       1.721613  0.869464  0.115094
764    -0.547562  0.034575       0.046215       0.405181 -0.692439  0.609757
765     0.342757  0.003299       0.149543       0.154433  0.279412 -0.734711
766    -0.844335  0.159683      -0.470426      -1.287373 -0.692439 -0.240048
767    -0.844335 -0.872451       0.046215       0.655930 -0.692439 -0.201997

     DiabetesPedigreeFunction       Age
0                    0.468187  1.425067
1                   -0.364823 -0.190548
2                    0.604004 -0.105515
```

```
3                   -0.920163 -1.040871
4                    5.481337 -0.020483
..                        …        …
763                 -0.908090  2.530487
764                 -0.398023 -0.530677
765                 -0.684747 -0.275580
766                 -0.370859  1.169970
767                 -0.473476 -0.870806

[768 rows x 8 columns]
```

[ ]:
```
# splitting X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.4,␣
 ↪random_state=37)

# training the model on training set
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# making predictions on the testing set
y_pred = gnb.predict(X_test)

# comparing actual response values (y_test) with predicted response values␣
 ↪(y_pred)
from sklearn import metrics
print("Gaussian Naive Bayes model accuracy(in %):", metrics.
 ↪accuracy_score(y_test, y_pred)*100)
```

```
Gaussian Naive Bayes model accuracy(in %): 77.59740259740259
```

0.2 Regression using a Regression Tree

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

df = pd.read_csv("/content/Walmart_sales.csv")
df
```

[ ]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	\
0	1	05-02-2010	1643690.90	0	42.31	2.572	
1	1	12-02-2010	1641957.44	1	38.51	2.548	
2	1	19-02-2010	1611968.17	0	39.93	2.514	
3	1	26-02-2010	1409727.59	0	46.63	2.561	
4	1	05-03-2010	1554806.68	0	46.50	2.625	
…	…	…	…	…	…	…	

```
6430     45  28-09-2012     713173.95             0        64.88       3.997
6431     45  05-10-2012     733455.07             0        64.89       3.985
6432     45  12-10-2012     734464.36             0        54.47       4.000
6433     45  19-10-2012     718125.53             0        56.47       3.969
6434     45  26-10-2012     760281.43             0        58.85       3.882

              CPI  Unemployment
0      211.096358         8.106
1      211.242170         8.106
2      211.289143         8.106
3      211.319643         8.106
4      211.350143         8.106
…             …             …
6430   192.013558         8.684
6431   192.170412         8.667
6432   192.327265         8.667
6433   192.330854         8.667
6434   192.308899         8.667

[6435 rows x 8 columns]
```

[ ]:
```
X = df.drop(["Date"], axis=1)
X
```

[ ]:
```
       Store  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price         CPI  \
0          1    1643690.90             0        42.31       2.572  211.096358
1          1    1641957.44             1        38.51       2.548  211.242170
2          1    1611968.17             0        39.93       2.514  211.289143
3          1    1409727.59             0        46.63       2.561  211.319643
4          1    1554806.68             0        46.50       2.625  211.350143
…      …            …             …            …          …           …
6430      45     713173.95             0        64.88       3.997  192.013558
6431      45     733455.07             0        64.89       3.985  192.170412
6432      45     734464.36             0        54.47       4.000  192.327265
6433      45     718125.53             0        56.47       3.969  192.330854
6434      45     760281.43             0        58.85       3.882  192.308899

       Unemployment
0             8.106
1             8.106
2             8.106
3             8.106
4             8.106
…             …
6430          8.684
6431          8.667
6432          8.667
```

```
6433         8.667
6434         8.667

[6435 rows x 7 columns]
```

[ ]:
```
y = df["Weekly_Sales"]
y
```

[ ]:
```
0       1643690.90
1       1641957.44
2       1611968.17
3       1409727.59
4       1554806.68
           …
6430     713173.95
6431     733455.07
6432     734464.36
6433     718125.53
6434     760281.43
Name: Weekly_Sales, Length: 6435, dtype: float64
```

[ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=37)
```

[ ]:
```
from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor(random_state=44)
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

[ ]:
```
print(predictions)
```

```
[2077256.24  907262.47  723708.99 … 1781528.77 1509323.09  813756.09]
```

[ ]:
```
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(y_test, predictions)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 10645746.83003515
```