220962050_Arhaan_Lab03

August 9, 2024

1 Lab 03

1.0.1 Question 1

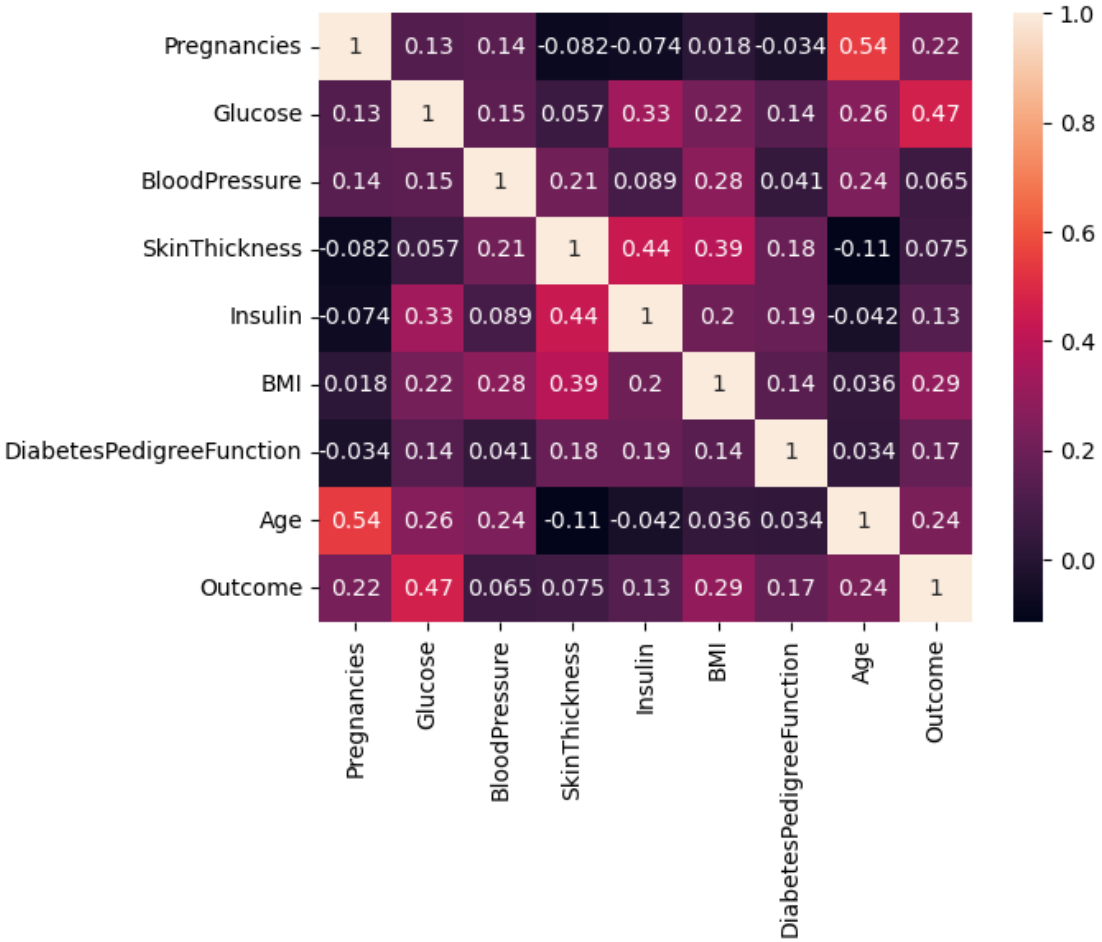
[ ]:
```
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
```

[ ]:
```
#! 1. Load Data in Pandas

data = pd.read_csv("diabetes_csv.csv")
data
```

[ ]:
```
     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]
```

[ ]:
```
corr_matrix = data.corr()
sn.heatmap(corr_matrix, annot=True)
plt.show()
```



[ ]:
```
#! 2. Drop Useless Columns

df = pd.read_csv("diabetes_csv.csv")
df.drop(labels=['BloodPressure'], axis=1, inplace=True)
df
```

[ ]:
```
     Pregnancies  Glucose  SkinThickness  Insulin   BMI  \
0              6      148             35        0  33.6
1              1       85             29        0  26.6
```

```
2              8      183              0        0  23.3
3              1       89             23       94  28.1
4              0      137             35      168  43.1
..           …      …              …       …   …
763           10      101             48      180  32.9
764            2      122             27        0  36.8
765            5      121             23      112  26.2
766            1      126              0        0  30.1
767            1       93             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 8 columns]
```

[ ]:
```
#! 3. Drop Rows w/ Missing Values

df = df.dropna()
df
```

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

[ ]:
```
#! 4. Create Dummy Variables

import random
df['Name'] = 'abc'
df.at[3,'Name']=np.nan
df

#! 5. Taking Care of Missing Data

df = df.dropna()
df
```

[ ]:
```
     Pregnancies  Glucose  SkinThickness  Insulin   BMI  \
0              6      148             35        0  33.6
1              1       85             29        0  26.6
2              8      183              0        0  23.3
4              0      137             35      168  43.1
5              5      116              0        0  25.6
..           …      …              …       …   …
763           10      101             48      180  32.9
764            2      122             27        0  36.8
765            5      121             23      112  26.2
766            1      126              0        0  30.1
767            1       93             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome Name
0                       0.627   50        1  abc
1                       0.351   31        0  abc
2                       0.672   32        1  abc
4                       2.288   33        1  abc
5                       0.201   30        0  abc
..                        …   …       …  …
763                     0.171   63        0  abc
764                     0.340   27        0  abc
765                     0.245   30        0  abc
766                     0.349   47        1  abc
```

```
767                    0.315   23        0  abc

[767 rows x 9 columns]
```

[ ]:
```python
#! 6.Convert the dataframe to numpy

from sklearn.model_selection import train_test_split

# Display the first few rows of the dataframe
print(df.head())

# features and target variable
X = df['Age']
y = df['Outcome']

#! 7.Divide the dataset into training and test data

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=104,␣
 ↪test_size=0.25, shuffle=True)
print('X_train : ')
print(X_train.head())
print('')
print('X_test : ')
print(X_test.head())
print('')
print('y_train : ')
print(y_train.head())
print('')
print('y_test : ')
print(y_test.head())
```

```
   Pregnancies  Glucose  SkinThickness  Insulin   BMI  \
0            6      148             35        0  33.6
1            1       85             29        0  26.6
2            8      183              0        0  23.3
4            0      137             35      168  43.1
5            5      116              0        0  25.6

   DiabetesPedigreeFunction  Age  Outcome Name
0                     0.627   50        1  abc
1                     0.351   31        0  abc
2                     0.672   32        1  abc
4                     2.288   33        1  abc
5                     0.201   30        0  abc
X_train :
199    29
750    22
140    55
```

```
728    22
38     27
Name: Age, dtype: int64

X_test :
119    21
154    43
176    42
477    31
630    34
Name: Age, dtype: int64

y_train :
199    1
750    1
140    0
728    0
38     1
Name: Outcome, dtype: int64

y_test :
119    0
154    1
176    0
477    0
630    1
Name: Outcome, dtype: int64
```

1.0.2 Question 2

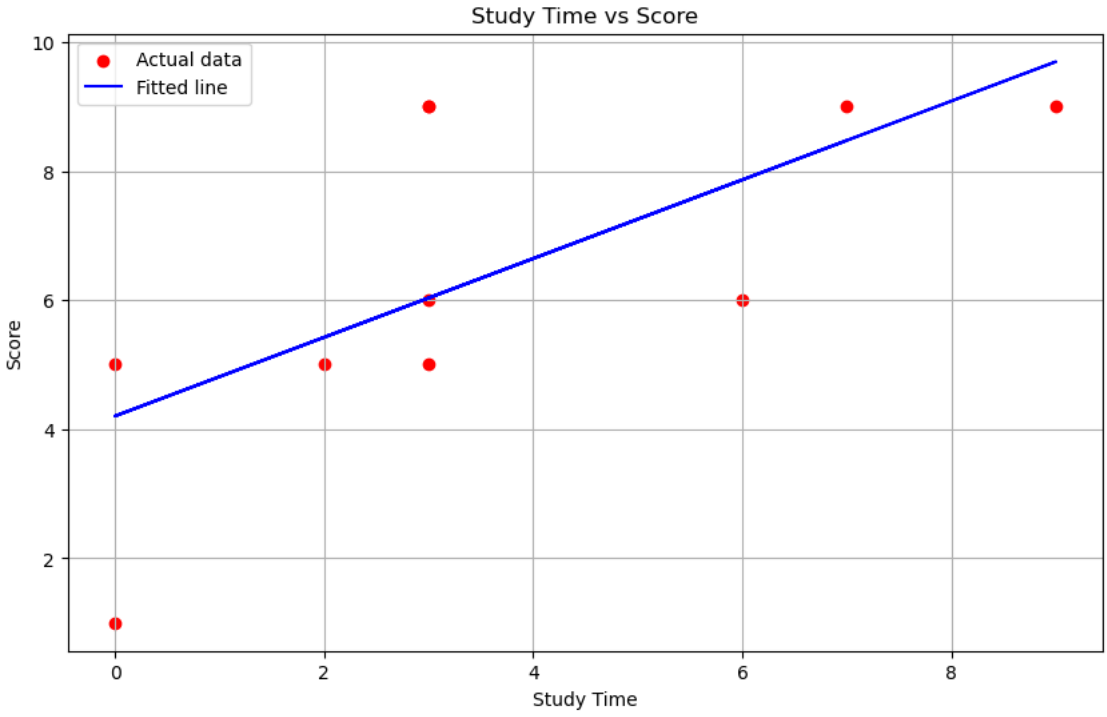
[ ]:
```
import csv
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[ ]:
```
'''
a. Construct a CSV file with the following attributes:
Study time in hours of ML lab course (x)
Score out of 10 (y)
The dataset should contain 10 rows.

b. Create a regression model and display the following:
Coefficients: B0 (intercept) and B1 (slope)
RMSE (Root Mean Square Error)
Predicted responses
```

```
c. Create a scatter plot of the data points in red color and plot the graph of␣
 ↪x vs. predicted y in blue color.
'''

data = [
    {'StudyTime': 3, 'Score': 5},
    {'StudyTime': 3, 'Score': 6},
    {'StudyTime': 0, 'Score': 5},
    {'StudyTime': 9, 'Score': 9},
    {'StudyTime': 0, 'Score': 1},
    {'StudyTime': 7, 'Score': 9},
    {'StudyTime': 3, 'Score': 9},
    {'StudyTime': 2, 'Score': 5},
    {'StudyTime': 6, 'Score': 6},
    {'StudyTime': 3, 'Score': 9},
]

with open('records.csv', 'w', newline='') as csvfile:
    fieldnames = ['StudyTime', 'Score']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    writer.writeheader()
    writer.writerows(data)

df = pd.read_csv("records.csv")

x = df['StudyTime'].values
y = df['Score'].values

x_mean = np.mean(x)
y_mean = np.mean(y)
xy_mean = np.mean(x * y)
x_sq_mean = np.mean(x ** 2)

B1 = np.sum((x - x_mean) * (y - y_mean)) / np.sum((x - x_mean) ** 2)
B0 = y_mean - B1 * x_mean

y_pred = B0 + B1 * x

rmse = np.sqrt(np.mean((y - y_pred) ** 2))

print(f"B0 (Intercept): {B0}")
print(f"B1 (Slope): {B1}")
print(f"RMSE: {rmse}")

plt.figure(figsize=(10, 6))
plt.scatter(x, y, color='red', label='Actual data')
plt.plot(x, y_pred, color='blue', label='Fitted line')
```

```
plt.xlabel('Study Time')
plt.ylabel('Score')
plt.title('Study Time vs Score')
plt.legend()
plt.grid(True)
plt.show()
```

```
B0 (Intercept): 4.204188481675393
B1 (Slope): 0.6099476439790575
RMSE: 1.8432699148680294
```



[ ]:
```
"""
!   2d.
Implement the model using two methods:
    Pedhazur formula (intuitive)
    Calculus method (partial derivatives, refer to class notes)
"""

N = len(x)
sum_x = np.sum(x)
sum_y = np.sum(y)
sum_xy = np.sum(x * y)
sum_x2 = np.sum(x ** 2)
```

```
B1_intuitive = (N * sum_xy - sum_x * sum_y) / (N * sum_x2 - sum_x ** 2)
B0_intuitive = (sum_y - B1_intuitive * sum_x) / N

print(f"Intercept (Pedhazur B0): {B0_intuitive}")
print(f"Slope (Pedhazur B1): {B1_intuitive}")
```

```
Intercept (Pedhazur B0): 4.204188481675393
Slope (Pedhazur B1): 0.6099476439790575
```

[ ]:
```
learning_rate = 0.01
epochs = 1000

B0_calc, B1_calc = 0, 0

for _ in range(epochs):
    y_pred_calc = B0_calc + B1_calc * x
    error = y_pred_calc - y
    B0_calc -= learning_rate * (2 / len(x)) * np.sum(error)
    B1_calc -= learning_rate * (2 / len(x)) * np.sum(error * x)

print(f"Intercept (Calculus B0): {B0_calc}")
print(f"Slope (Calculus B1): {B1_calc}")
```

```
Intercept (Calculus B0): 4.2012827719909795
Slope (Calculus B1): 0.6104644619857059
```

[ ]:
```
"""
    !2e.
Compare the coefficients obtained using both methods and compare them with the␣
 ↪analytical solution"""

print(f"Pedhazur Intercept (B0): {B0_intuitive}")
print(f"Pedhazur Slope (B1): {B1_intuitive}")
print(f"Calculus Intercept (B0): {B0_calc}")
print(f"Calculus Slope (B1): {B1_calc}")
```

```
Pedhazur Intercept (B0): 4.204188481675393
Pedhazur Slope (B1): 0.6099476439790575
Calculus Intercept (B0): 4.2012827719909795
Calculus Slope (B1): 0.6104644619857059
```

[ ]:
```
"""
    !2f.
Test your model to predict the score obtained when the study time of a student␣
 ↪is 10 hours.
"""

study_time_test = 10
```

```
score_pred = B0 + B1 * study_time_test
print(f"Predicted score for 10 hours of study time: {score_pred}")
```

```
Predicted score for 10 hours of study time: 10.30366492146597
```