WEEK_3

August 7, 2024

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

[ ]:
```
# 1) Consider the hepatitis/ pima-indians-diabetes csv file, perform the␣
↪following date pre-processing.
```

[41]:
```
# 1. Load data in Pandas.
df = pd.read_csv('diabetes_csv.csv')
df
```

[41]:

	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]

[42]:
```
# 2. Drop columns that aren't useful.
df.drop(['Outcome', 'Age'], axis=1, inplace=True)
```

[43]:
```
df
```

[43]:
```
     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
0                       0.627
1                       0.351
2                       0.672
3                       0.167
4                       2.288
..                        ...
763                     0.171
764                     0.340
765                     0.245
766                     0.349
767                     0.315

[768 rows x 7 columns]
```

[44]:
```
# 3. Drop rows with missing values.
df.dropna(inplace=True)
```

[45]:
```
df
```

[45]:
```
     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
0                       0.627
1                       0.351
2                       0.672
3                       0.167
4                       2.288
..                        …
763                     0.171
764                     0.340
765                     0.245
766                     0.349
767                     0.315

[768 rows x 7 columns]
```

[72]:
```
# 4. Create dummy variables.
df_dummies = pd.get_dummies(df, columns=['Insulin'])
df_dummies
```

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

     DiabetesPedigreeFunction  Insulin_0  Insulin_14  Insulin_15  Insulin_16  \
0                       0.627       True       False       False       False
1                       0.351       True       False       False       False
2                       0.672       True       False       False       False
3                       0.167      False       False       False       False
4                       2.288      False       False       False       False
..                        …          …          …           …           …
763                     0.171      False       False       False       False
764                     0.340       True       False       False       False
765                     0.245      False       False       False       False
766                     0.349       True       False       False       False
```

```
767                    0.315       True       False       False       False

     …  Insulin_495  Insulin_510  Insulin_540  Insulin_543  Insulin_545  \
0    …        False        False        False        False        False
1    …        False        False        False        False        False
2    …        False        False        False        False        False
3    …        False        False        False        False        False
4    …        False        False        False        False        False
..   …          …            …            …            …            …
763  …        False        False        False        False        False
764  …        False        False        False        False        False
765  …        False        False        False        False        False
766  …        False        False        False        False        False
767  …        False        False        False        False        False

     Insulin_579  Insulin_600  Insulin_680  Insulin_744  Insulin_846
0          False        False        False        False        False
1          False        False        False        False        False
2          False        False        False        False        False
3          False        False        False        False        False
4          False        False        False        False        False
..           …            …            …            …            …
763        False        False        False        False        False
764        False        False        False        False        False
765        False        False        False        False        False
766        False        False        False        False        False
767        False        False        False        False        False

[768 rows x 192 columns]
```

[48]:
```
# 5. Take care of missing data.
df = df.fillna(df.median())
df
```

[48]:
```
     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
0                       0.627
1                       0.351
2                       0.672
3                       0.167
4                       2.288
..                        …
763                     0.171
764                     0.340
765                     0.245
766                     0.349
767                     0.315

[768 rows x 7 columns]
```

[73]:
```
# 6. Convert the data frame to NumPy.
data = df.to_numpy()
data
```

[73]:
```
array([[  6.   , 148.   ,  72.   , …,   0.   ,  33.6  ,   0.627],
       [  1.   ,  85.   ,  66.   , …,   0.   ,  26.6  ,   0.351],
       [  8.   , 183.   ,  64.   , …,   0.   ,  23.3  ,   0.672],
       …,
       [  5.   , 121.   ,  72.   , …, 112.   ,  26.2  ,   0.245],
       [  1.   , 126.   ,  60.   , …,   0.   ,  30.1  ,   0.349],
       [  1.   ,  93.   ,  70.   , …,   0.   ,  30.4  ,   0.315]])
```

[55]:
```

```

[57]:
```

```

```
{'Pregnancies': <class 'numpy.int64'>, 'Glucose': <class 'numpy.int64'>,
'BloodPressure': <class 'numpy.int64'>, 'SkinThickness': <class 'numpy.int64'>,
'Insulin': <class 'numpy.int64'>, 'BMI': <class 'numpy.float64'>,
'DiabetesPedigreeFunction': <class 'numpy.float64'>}
```

[74]:
```
# 7. Divide the data set into training data and test data.
import pandas as pd
import numpy as np

df = pd.read_csv('diabetes_csv.csv')

train_proportion = 0.8  # 80% for training and 20% for testing

shuffled_df = df.sample(frac=1, random_state=42).reset_index(drop=True)

split_index = int(train_proportion * len(shuffled_df))
```

```
train_df = shuffled_df[:split_index]
test_df = shuffled_df[split_index:]

X_train = train_df.drop(columns=['Outcome'])
y_train = train_df['Outcome']

X_test = test_df.drop(columns=['Outcome'])
y_test = test_df['Outcome']

print("Training features:\n", X_train.head())
print("Training targets:\n", y_train.head())
print("Test features:\n", X_test.head())
print("Test targets:\n", y_test.head())
```

```
Training features:
    Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6       98             58             33      190  34.0
1            2      112             75             32        0  35.7
2            2      108             64              0        0  30.8
3            8      107             80              0        0  24.6
4            7      136             90              0        0  29.9

   DiabetesPedigreeFunction  Age
0                     0.430   43
1                     0.148   21
2                     0.158   21
3                     0.856   34
4                     0.210   50
Training targets:
 0    0
1    0
2    0
3    0
4    0
Name: Outcome, dtype: int64
Test features:
      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
614            3      116              0              0        0  23.5
615           10       75             82              0        0  33.3
616            7      137             90             41        0  32.0
617            3      158             64             13      387  31.2
618            7      129             68             49      125  38.5

     DiabetesPedigreeFunction  Age
614                     0.187   23
615                     0.263   38
616                     0.391   39
617                     0.295   24
```

```
618                     0.439   43
Test targets:
 614    0
615    0
616    0
617    0
618    1
Name: Outcome, dtype: int64
```

[ ]:
```
# 2.
# 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.

# d. Implement the model using two methods:
# Pedhazur formula (intuitive)
# Calculus method (partial derivatives, refer to class notes)

# e. Compare the coefficients obtained using both methods and compare them with␣
 ↪the analytical solution.

# f. Test your model to predict the score obtained when the study time of a␣
 ↪student is 10 hours.
```

[92]:
```
import numpy as np
import pandas as pd

df = pd.read_csv('study.csv')

X = df['Study Time'].values
y = df['Score'].values

X_matrix = np.vstack([np.ones(len(X)), X]).T

# Calculate coefficients using the normal equation
coefficients = np.linalg.inv(X_matrix.T @ X_matrix) @ X_matrix.T @ y
B0, B1 = coefficients
```

```
# Calculate predictions
predictions = X_matrix @ coefficients

# Calculate RMSE
rmse = np.sqrt(np.mean((y - predictions) ** 2))

print(f'Intercept (B0): {B0}')
print(f'Slope (B1): {B1}')
print(f'RMSE: {rmse}')
print(f'Predicted responses: {predictions}')
```

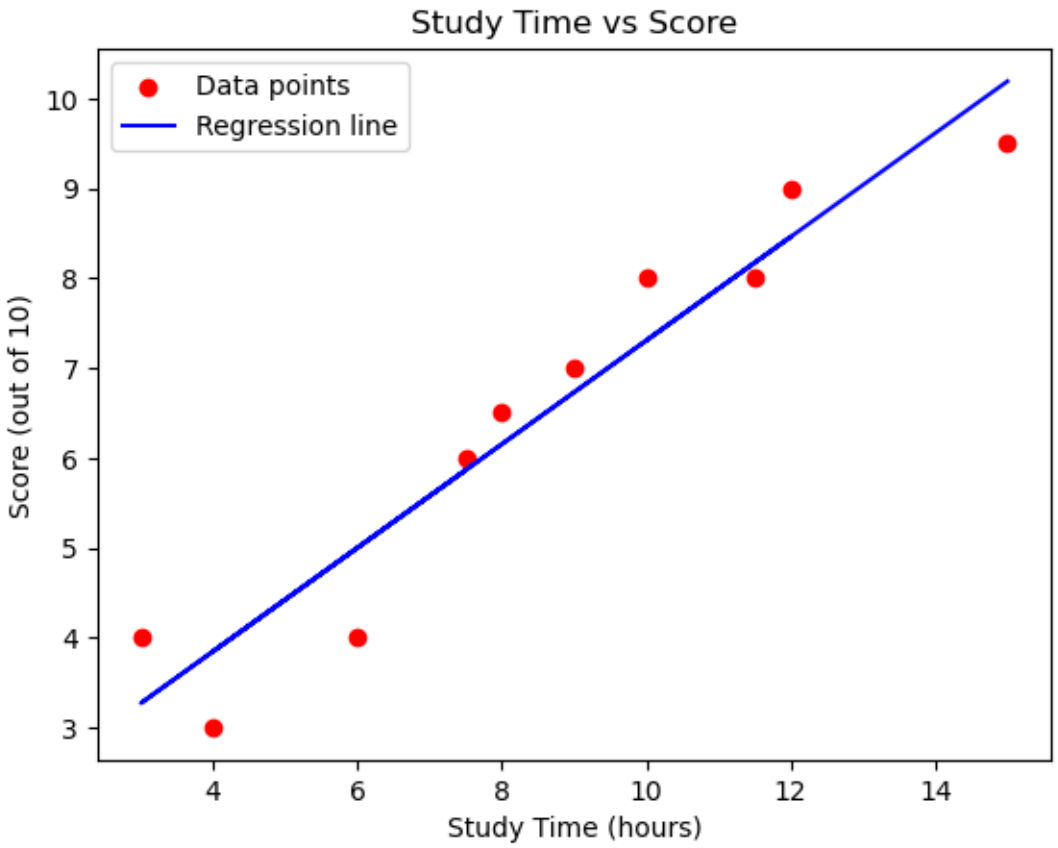
```
Intercept (B0): 1.537126715092815
Slope (B1): 0.5770782889426964
RMSE: 0.6114656442186477
Predicted responses: [ 7.3079096   8.46206618  8.17352704  3.84543987
4.99959645  5.86521388
  3.26836158  6.15375303  6.73083132 10.19330105]
```

[93]:
```
import matplotlib.pyplot as plt

# Scatter plot of original data
plt.scatter(X, y, color='red', label='Data points')

# Plot the regression line
plt.plot(X, predictions, color='blue', label='Regression line')

plt.xlabel('Study Time (hours)')
plt.ylabel('Score (out of 10)')
plt.title('Study Time vs Score')
plt.legend()
plt.show()
```



[88]:
```
# Calculate means
mean_X = np.mean(X)
mean_y = np.mean(y)

# Calculate B1
B1_ped = np.sum((X - mean_X) * (y - mean_y)) / np.sum((X - mean_X) ** 2)

# Calculate B0
B0_ped = mean_y - B1_ped * mean_X

print(f'Pedhazur Formula - Intercept (B0): {B0_ped}')
print(f'Pedhazur Formula - Slope (B1): {B1_ped}')
```

```
Pedhazur Formula - Intercept (B0): 1.8452380952380958
Pedhazur Formula - Slope (B1): 0.5238095238095237
```

[89]:
```
# Coefficients using normal equations
coefficients_calc = np.linalg.inv(X_matrix.T @ X_matrix) @ X_matrix.T @ y
B0_calc, B1_calc = coefficients_calc
```

```
print(f'Calculus Method - Intercept (B0): {B0_calc}')
print(f'Calculus Method - Slope (B1): {B1_calc}')
```

```
Calculus Method - Intercept (B0): 1.845238095238094
Calculus Method - Slope (B1): 0.5238095238095245
```

[90]:
```
print(f'Pedhazur Formula - Intercept (B0): {B0_ped}')
print(f'Pedhazur Formula - Slope (B1): {B1_ped}')
print(f'Calculus Method - Intercept (B0): {B0_calc}')
print(f'Calculus Method - Slope (B1): {B1_calc}')
print(f'Analytical Solution - Intercept (B0): {B0}')
print(f'Analytical Solution - Slope (B1): {B1}')
```

```
Pedhazur Formula - Intercept (B0): 1.8452380952380958
Pedhazur Formula - Slope (B1): 0.5238095238095237
Calculus Method - Intercept (B0): 1.845238095238094
Calculus Method - Slope (B1): 0.5238095238095245
Analytical Solution - Intercept (B0): 1.845238095238094
Analytical Solution - Slope (B1): 0.5238095238095245
```

[91]:
```
# Predict score for study time of 10 hours using the regression model
study_time = 10
predicted_score = B0 + B1 * study_time
print(f'Predicted score for {study_time} hours of study: {predicted_score}')
```

```
Predicted score for 10 hours of study: 7.083333333333339
```

[ ]:
```

```