Abhishek Murthy
21BDS0064
Fall Sem 2024-2025
DA - 2
Machine Learning Lab
21-08-2024

Linear Regression:

Dataset:

Time (s)	5	7	12	16	20
Mass (g)	40	120	180	210	240

Code:

```
df = [
   [5, 7, 12, 16, 20],
   [40, 120, 180, 210, 240]
]
n = len(df[0])
sigma_x = sum(df[0])
sigma_y = sum(df[1])
sigma_xy = sum([x*y for x, y in zip(df[0], df[1])])
sigma_x_square = sum([x**2 for x in df[0]])

a = (sigma_y*sigma_x_square - sigma_x*sigma_xy) / (n*sigma_x_square - sigma_x**2)
b = (n*sigma_xy - sigma_x*sigma_y) / (n*sigma_x_square - sigma_x**2)

x = 10
y = a + b*x
print(y)
```

Output:

```
a = (sigma_y*sigma_x_square - sigma_x*sigma_xy) / (n*sigma_x_square - sigma_x**2)
b = (n*sigma_xy - sigma_x*sigma_y) / (n*sigma_x_square - sigma_x**2)
print(a,b)
```
✓ 0.0s

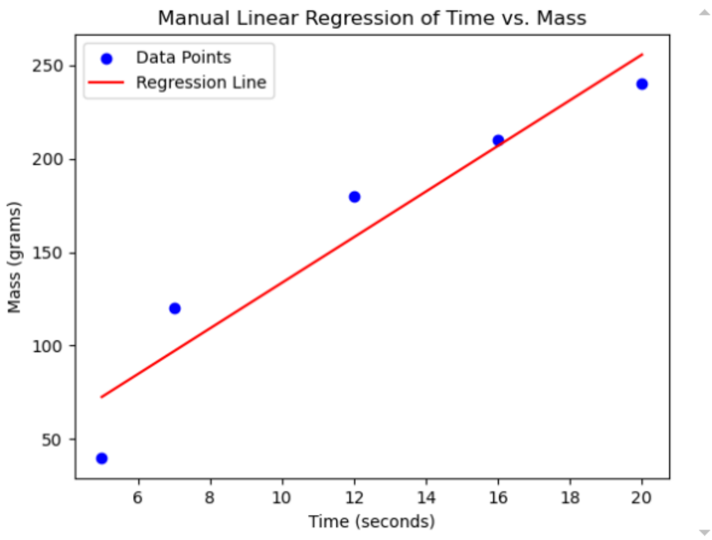
```
11.506493506493506 12.207792207792208
```

```
x = 10
y = a + b*x
print(y)
```
✓ 0.0s

```
133.58441558441558
```

Plot:

```
import matplotlib.pyplot as plt
time_line = df[0]
mass_line = [a + b * x for x in time_line]

plt.scatter(df[0], df[1], color='blue', label='Data Points')
plt.plot(time_line, mass_line, color='red', label='Regression Line')

plt.xlabel('Time (seconds)')
plt.ylabel('Mass (grams)')
plt.title('Manual Linear Regression of Time vs. Mass')
plt.legend()
plt.show()
```



Multiple Linear Regression:

Dataset:

Salary	Education	Experience
30000	11	10
27000	11	6
20000	12	10
25000	12	5
29000	13	5
35000	14	6
38000	14	5
40000	16	8
45000	16	7
28000	16	2
30000	18	6
55000	18	2
65000	22	5
25000	23	2
75000	24	1

Code:

```
df = [
   [30000, 27000, 20000, 25000, 29000, 35000, 38000, 40000, 45000, 28000,
30000, 55000, 65000, 25000],
   [11, 11, 12, 12, 13, 14, 14, 16, 16, 16, 18, 18, 22, 23],
   [10, 6, 10, 5, 5, 6, 5, 8, 7, 2, 6, 2, 5, 2]
]
n = len(df[0])
sum_y = sum(df[0])
mean_y = sum_y/n

sum_x1 = sum(df[1])
mean_x1 = sum_x1/n
sum_x2 = sum(df[2])
mean_x2 = sum_x2/n
print(sum_x1, sum_x2, sum_y, mean_y, mean_x1, mean_x2)
x1_square = [i**2 for i in df[1]]
sum_x1_square = sum(x1_square)

x2_square = [i**2 for i in df[2]]
```

```
sum_x2_square = sum(x2_square)

y2 = [i**2 for i in df[0]]
x1y = [df[1][i]*df[0][i] for i in range(len(df[0]))]
sum_x1y = sum(x1y)
x2y = [df[2][i]*df[0][i] for i in range(len(df[0]))]
sum_x2y = sum(x2y)
x1x2 = [df[1][i]*df[2][i] for i in range(len(df[0]))]
sum_x1x2 = sum(x1x2)

sigmax1_square = sum_x1_square - sum_x1**2/n
sigmax2_square = sum_x2_square - sum_x2**2/n
sigma_x1y = sum_x1y - sum_x1*sum_y/n
sigma_x2y = sum_x2y - sum_x2*sum_y/n
sigma_x1x2 = sum_x1x2 - sum_x1*sum_x2/n
b1 = (sigmax2_square*sigma_x1y -
sigma_x1x2*sigma_x2y)/(sigmax1_square*sigmax2_square - sigma_x1x2**2)
b2 = (sigmax1_square*sigma_x2y - sigma_x1x2*sigma_x1y)
/(sigmax1_square*sigmax2_square - sigma_x1x2**2)
print(b1, b2)

b0 = mean_y - b1*mean_x1 - b2*mean_x2

X1 = 60
X2 = 12
Y_pred = b0 + b1*X1 + b2*X2
```

Output:

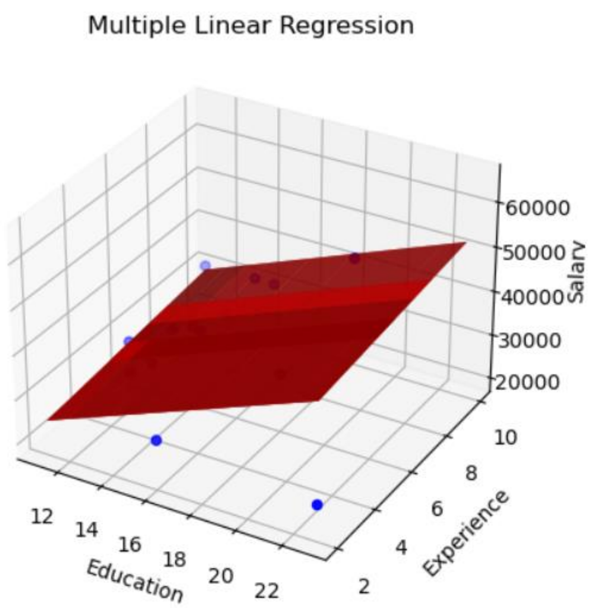
```
X1 = 60
X2 = 12
Y_pred = b0 + b1*X1 + b2*X2
```
4] ✓ 0.0s Python

```
Y_pred
```
5] ✓ 0.0s Python

```
124035.1617889399
```

Plot:

```
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import matplotlib.pyplot as plt

x1_grid, x2_grid = np.meshgrid(df[1], df[2])
y_grid = b0 + b1 * x1_grid + b2 * x2_grid

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Scatter plot for original data points
ax.scatter(df[1], df[2], df[0], color='blue', label='Data Points')

# Plotting the regression plane
ax.plot_surface(x1_grid, x2_grid, y_grid, color='red', alpha=0.5,
label='Regression Plane')

# Labels and title
ax.set_xlabel('Education')
ax.set_ylabel('Experience')
ax.set_zlabel('Salary')
ax.set_title('Multiple Linear Regression')

plt.show()
```



Naïve Bayes

Dataset

classification.csv

```
Color,Engine Type,Top Speed,Aerodynamics,Team
Red,Hybrid,Fast,Good,Red Bull
Blue,Electric,Medium,Excellent,Mercedes
Blue,Hybrid,Fast,Fair,Red Bull
Red,Hybrid,Medium,Good,Red Bull
Blue,Electric,Fast,Fair,Mercedes
Red,Hybrid,Medium,Excellent,Mercedes
Blue,Electric,Fast,Good,Red Bull
Red,Hybrid,Fast,Excellent,Red Bull
```

Code:

```
import pandas as pd
from collections import Counter
df = pd.read_csv('classification.csv')

def class_probs(df, target):
   total = len(df)
   class_counts = Counter(df[target])
   class_probs = {i: ct / total for i, ct in class_counts.items()}
   return class_counts, class_probs

def feature_probs(df, feature, target):
   feature_dict = {}
   for class_ in df[target].unique():
      mini_df = df[df[target] == class_]
      feature_counts = Counter(mini_df[feature].astype(str))
      tot_count = len(mini_df)
      feature_dict[class_] = {f"{val}": count / tot_count for val, count in
feature_counts.items()}

   return feature_dict

def calc_probs(instance, feat_probs, class_probs):
   inst_probs = {}
   for class_, class_prob in class_probs.items():
```

```
        probs = class_prob
        for i, feature_val in enumerate(instance):
            if feature_val in feat_probs[i][class_]:
                probs *= feat_probs[i][class_][feature_val]
            else:
                probs *= 0
        inst_probs[class_] = probs
    return inst_probs


prediction = "Red,Electric,Fast,Good"
prediction = list(prediction.split(","))
val = calc_probs(prediction, feature_probs_list, class_prob)
final_class = max(val, key=val.get)
for i, j in val.items():
    print(f"{i}: {j}")
```

Output:

```
prediction = "Red,Electric,Fast,Good"          Add to chat (Ctrl+L) | Edit
prediction = list(prediction.split(","))
val = calc_probs(prediction, feature_probs_list, class_prob)
final_class = max(val, key=val.get)
for i, j in val.items():
    print(f"{i}: {j}")
```

✓ 0.0s

```
Red Bull: 0.036000000000000004
Mercedes: 0.0
```

```
print(f"The final prediction for the given input is: {final_class}")
```

✓ 0.0s

```
The final prediction for the given input is: Red Bull
```