assignment2-mml

October 4, 2025

Assignment 2 AIT 512 Maths for ML * Rollno. MT2025001 * Name: Aayank Singhai * Question 1 Least Square Classifier

[107]:
```
!pip install ucimlrepo
```

```
Requirement already satisfied: ucimlrepo in /usr/local/lib/python3.12/dist-
packages (0.0.7)
Requirement already satisfied: pandas>=1.0.0 in /usr/local/lib/python3.12/dist-
packages (from ucimlrepo) (2.2.2)
Requirement already satisfied: certifi>=2020.12.5 in
/usr/local/lib/python3.12/dist-packages (from ucimlrepo) (2025.8.3)
Requirement already satisfied: numpy>=1.26.0 in /usr/local/lib/python3.12/dist-
packages (from pandas>=1.0.0->ucimlrepo) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.12/dist-packages (from pandas>=1.0.0->ucimlrepo)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-
packages (from pandas>=1.0.0->ucimlrepo) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-
packages (from pandas>=1.0.0->ucimlrepo) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-
packages (from python-dateutil>=2.8.2->pandas>=1.0.0->ucimlrepo) (1.17.0)
```

[108]:
```
#Importing required packages
import numpy as np
import pandas as pd
from ucimlrepo import fetch_ucirepo
from sklearn.metrics import confusion_matrix, classification_report
```

[109]:
```
iris_dts = fetch_ucirepo(id=53)
data = iris_dts.data
```

[110]:
```
#Loading data variables
x = data.features
y = data.targets
```

[111]:
```
x_values = x.values # Mat Dimens 150x4
y_values = y.values.flatten() # Goal : (150,)
```

[112]:

```
x = x_values.T #Transposing the x mat
print(f"The Shape of the matrix (data) of x is : {x.shape}")



y_true = y_values.astype(str)
y = np.where(y_true == 'Iris-setosa', 1, -1).reshape(-1, 1)
print(f"The Shape of the label vector <y> is : {y.shape}")
```

```
The Shape of the matrix (data) of x is : (4, 150)
The Shape of the label vector <y> is : (150, 1)
```

[113]:

```
# Creating a regression model
colnms_ones = np.ones((x.shape[1], 1))
x_augmn_Tp = np.hstack((x.T, colnms_ones))
x_augmn = x_augmn_Tp.T

# print(x_augmn)

print("\n<--- Model Params Below --->")
print(f"The Shape of the augmented matrix x_augmn_Tp is : {x_augmn_Tp.shape}")

# Regression -> Closed Form Equation
beta_til = np.linalg.inv(x_augmn @ x_augmn_Tp) @ x_augmn @ y

beta_val = beta_til[:-1] #beta value assignment
bias_coef = beta_til[-1] #bias val

print(f"The Calculated beta ( ) is: \n{beta_val}")
print(f"Final Calculated ( ) is : {bias_coef}")

y_pred = np.sign(x_augmn_Tp @ beta_til) #Regression model as classifier
```

```

<--- Model Params Below --->
The Shape of the augmented matrix x_augmn_Tp is : (150, 5)
The Calculated beta ( ) is:
[[ 0.1312861 ]
 [ 0.48494601]
 [-0.44552275]
 [-0.12670283]]
Final Calculated ( ) is : [-0.75506092]
```

[114]:

```
# Building Confusion Matrix
conf_mtrix = confusion_matrix(y, y_pred, labels=[1, -1])

conf_mtrix_df = pd.DataFrame(conf_mtrix,
                     index=['Actual: Setosa (+1)', 'Actual: Not Setosa (-1)'],
```

```
                    columns=['Predicted: Setosa (+1)', 'Predicted: Not Setosa␣
↪(-1)'])

print("The Confusion Matrix is:")
print(conf_mtrix_df)
print("\nConclusion: The Iris-setosa class is 'linearly separable'.")
print("This means it's so distinct that we could draw a single straight line")
print("to completely separate it from the other two flower types.")
```

```
The Confusion Matrix is:
                         Predicted: Setosa (+1)  Predicted: Not Setosa (-1)
Actual: Setosa (+1)                          50                           0
Actual: Not Setosa (-1)                       0                         100

Conclusion: The Iris-setosa class is 'linearly separable'.
This means it's so distinct that we could draw a single straight line
to completely separate it from the other two flower types.
```

[115]:
```
import plotly.express as px

#Visualizing the Confusion matrix using plotly
fig = px.imshow(conf_mtrix_df,
                text_auto=True,
                color_continuous_scale='Blues',
                labels=dict(x="Predicted Flower Type", y="Actual Flower Type",␣
↪color="Count")
               )

fig.update_layout(title_text='Confusion Matrix Heatmap', title_x=0.5)
fig.show()
```

Assignment 2 AIT 512 Maths for ML * Rollno. MT2025001 * Name: Aayank Singhai * Question 2 Least Square Classifier for a reduced dimension dataset

[116]:
```
!pip install ucimlrepo
```

```
Requirement already satisfied: ucimlrepo in /usr/local/lib/python3.12/dist-
packages (0.0.7)
Requirement already satisfied: pandas>=1.0.0 in /usr/local/lib/python3.12/dist-
packages (from ucimlrepo) (2.2.2)
Requirement already satisfied: certifi>=2020.12.5 in
/usr/local/lib/python3.12/dist-packages (from ucimlrepo) (2025.8.3)
Requirement already satisfied: numpy>=1.26.0 in /usr/local/lib/python3.12/dist-
packages (from pandas>=1.0.0->ucimlrepo) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.12/dist-packages (from pandas>=1.0.0->ucimlrepo)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-
```

```
packages (from pandas>=1.0.0->ucimlrepo) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-
packages (from pandas>=1.0.0->ucimlrepo) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-
packages (from python-dateutil>=2.8.2->pandas>=1.0.0->ucimlrepo) (1.17.0)
```

[117]:
```python
#Importing required packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from ucimlrepo import fetch_ucirepo
from sklearn.metrics import confusion_matrix, classification_report
```

[118]:
```python
iris_dts = fetch_ucirepo(id=53)
data = iris_dts.data
```

[119]:
```python
#Loading data variables
x = data.features
y = data.targets
```

[120]:
```python
x_values = x.values # Mat Dimens 150x4
y_values = y.values.flatten() # Goal : (150,)
```

[121]:
```python
x = x_values.T #Transposing the x mat
print(f"The Shape of the matrix (data) of x is : {x.shape}")


y_true = y_values.astype(str)
y = np.where(y_true == 'Iris-setosa', 1, -1).reshape(-1, 1)
print(f"The Shape of the label vector <y> is : {y.shape}")
```

```
The Shape of the matrix (data) of x is : (4, 150)
The Shape of the label vector <y> is : (150, 1)
```

[122]:
```python
# Performing SVD
U, s, VT = np.linalg.svd(x)
print(" The Result of SVD is below")
print(f"Singular Values =  {s.round(3)}")
print(f"The Variance by the first two singular values = {(s[0]**2 + s[1]**2) /
 ↪np.sum(s**2) * 100:.3f}%")
```

```
 The Result of SVD is below
Singular Values =  [95.951 17.723  3.469  1.879]
The Variance by the first two singular values = 99.837%
```

[123]:
```
# decided to reduce data to just 2 key dimensions.
k = 2

u_k = U[:, :k]
s_k = np.diag(s[:k])
vt_k = VT[:k, :]

print(f"Shape of the new, simplified U matrix: {u_k.shape}")
print(f"Shape of the new, simplified S matrix: {s_k.shape}")
print(f"Shape of the new, simplified VT matrix: {vt_k.shape}")
```

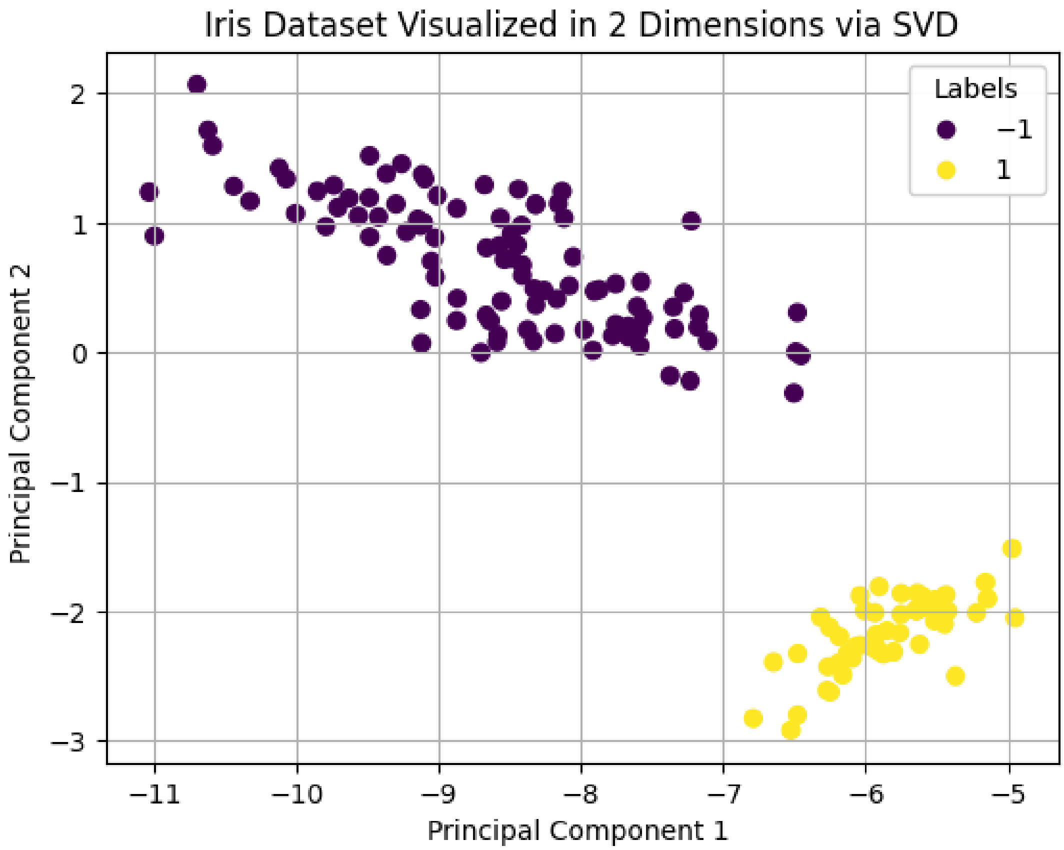
```
Shape of the new, simplified U matrix: (4, 2)
Shape of the new, simplified S matrix: (2, 2)
Shape of the new, simplified VT matrix: (2, 150)
```

[124]:
```
# X_hat calculation
X_hat = u_k @ s_k @ vt_k
# Reduced dataset: u_k^T X_hat (shape 2x150)
reduced = u_k.T @ X_hat

scatter = plt.scatter(reduced[0, :], reduced[1, :], c= y, cmap="viridis", s=40)
plt.grid(True)
plt.xlabel("Principal Component 1 ")
plt.ylabel("Principal Component 2 ")
plt.title("Iris Dataset Visualized in 2 Dimensions via SVD")

# Add legend
legend_1 = plt.legend(*scatter.legend_elements(), title="Labels")
plt.gca().add_artist(legend_1)

plt.show()
```



[125]:
```
ones = np.ones((X_hat.shape[1], 1))
X_hat_augmn_T = np.hstack((X_hat.T, ones))

# Get the transpose for the normal equation
X_hat_augmn = X_hat_augmn_T.T

beta_til_reduc = np.linalg.inv(X_hat_augmn @ X_hat_augmn_T) @ X_hat_augmn @ y


beta_val = beta_til_reduc[:-1] #beta value assignment
bias_coef = beta_til_reduc[-1] #bias val

print(f"The Calculated beta ( ) is: \n{beta_val}")
print(f"Final Calculated ( ) is : {bias_coef}")
```

```
The Calculated beta ( ) is:
[[ 0.1222063 ]
 [ 0.41560675]
 [-0.06695662]
 [-1.01516404]]
```

```
Final Calculated ( ) is : [-0.83036559]
```

[126]:
```python
import plotly.express as px

y_predicted_s = np.sign(X_hat_augmn_T @ beta_til_reduc)

cm_s = confusion_matrix(y, y_predicted_s, labels=[1, -1])
cm_s_df = pd.DataFrame(cm_s,
                       index=['Actual: Setosa (+1)', 'Not Setosa (-1)'],
                       columns=['Predicted: Setosa (+1)', 'Not Setosa (-1)'])

fig = px.imshow(cm_s_df,
                text_auto=True,
                color_continuous_scale='Blues',
                labels=dict(x="Predicted Label", y="Actual Label")
               )
fig.update_layout(
    title_text='Confusion Matrix (Reduced Dataset)',
    title_x=0.5
)
fig.update(layout_coloraxis_showscale=False)

fig.show()
```

[127]:
```python
print("--- Final Analysis ---")

print("\n1. Performance Comparison")
print("The simplified model, using only 2 dimensions, performed identically to␣
 ↪the full model. Both achieved perfect classification.")

print("\n2. Why Didn't Performance Decrease?")
print("The performance remained perfect because the first two principal␣
 ↪components found by SVD contained all the essential information needed to␣
 ↪separate the highly distinct Iris-setosa species. The discarded dimensions␣
 ↪were redundant for this specific task.")

print("\n3. How Does U  * X Reduce Dimension?")
print("Multiplying the data (X) by U  is a mathematical projection. It's like␣
 ↪casting a 2D shadow of a 3D object; the shadow (the new data) is a␣
 ↪lower-dimensional representation that captures the most important features.")
```

```
--- Final Analysis ---

1. Performance Comparison
The simplified model, using only 2 dimensions, performed identically to the full
model. Both achieved perfect classification.

2. Why Didn't Performance Decrease?
```

The performance remained perfect because the first two principal components found by SVD contained all the essential information needed to separate the highly distinct Iris-setosa species. The discarded dimensions were redundant for this specific task.

3. How Does U * X Reduce Dimension?
Multiplying the data (X) by U is a mathematical projection. It's like casting a 2D shadow of a 3D object; the shadow (the new data) is a lower-dimensional representation that captures the most important features.