Sudeshna Ghosh 23122134 3MSc.DS(B)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix,
ConfusionMatrixDisplay
from sklearn.decomposition import PCA

data = {
    'Pregnancies': [6, 1, 8, 1, 0, 5, 3, 10, 2, 8],
    'Glucose': [148, 85, 183, 89, 137, 116, 78, 115, 197, 125],
    'BloodPressure': [72, 66, 64, 66, 40, 74, 50, 0, 70, 96],
    'SkinThickness': [35, 29, 0, 23, 35, 0, 32, 0, 45, 0],
    'Insulin': [0, 0, 0, 94, 168, 0, 88, 0, 543, 0],
    'BMI': [33.6, 26.6, 23.3, 28.1, 43.1, 25.6, 31.0, 35.3, 30.5,
32.0],
    'DiabetesPedigreeFunction': [0.627, 0.351, 0.672, 0.167, 2.288,
0.201, 0.248, 0.134, 0.158, 0.232],
    'Age': [50, 31, 32, 21, 33, 30, 26, 29, 53, 54],
    'Outcome': [1, 0, 1, 0, 1, 0, 0, 0, 1, 1]
}

df = pd.DataFrame(data)

X = df.drop('Outcome', axis=1)
y = df['Outcome']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Feature scaling
scaler = StandardScaler()

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

knn = KNeighborsClassifier(n_neighbors=3)  # You can change the number
of neighbors
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
sample_data = [[6, 148, 72, 35, 0, 33.6, 0.627, 50]]
scaled_sample_data = scaler.transform(sample_data)
prediction = knn.predict(scaled_sample_data)
```

```
print(f'The prediction for the sample data is: {prediction[0]}')
```

```
The prediction for the sample data is: 1
```

```
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
site-packages/sklearn/base.py:493: UserWarning: X does not have valid
feature names, but StandardScaler was fitted with feature names
  warnings.warn(
```

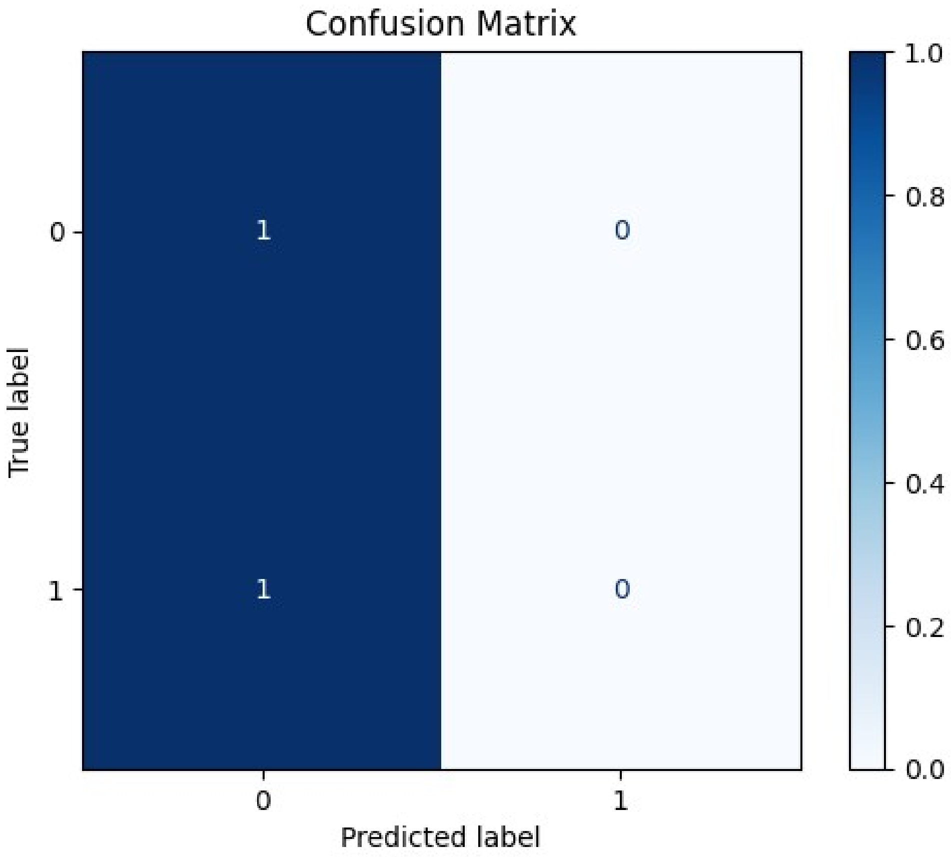
```
y_pred = knn.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[1 0]
 [1 0]]
              precision    recall  f1-score   support

           0       0.50      1.00      0.67         1
           1       0.00      0.00      0.00         1

    accuracy                           0.50         2
   macro avg       0.25      0.50      0.33         2
weighted avg       0.25      0.50      0.33         2
```

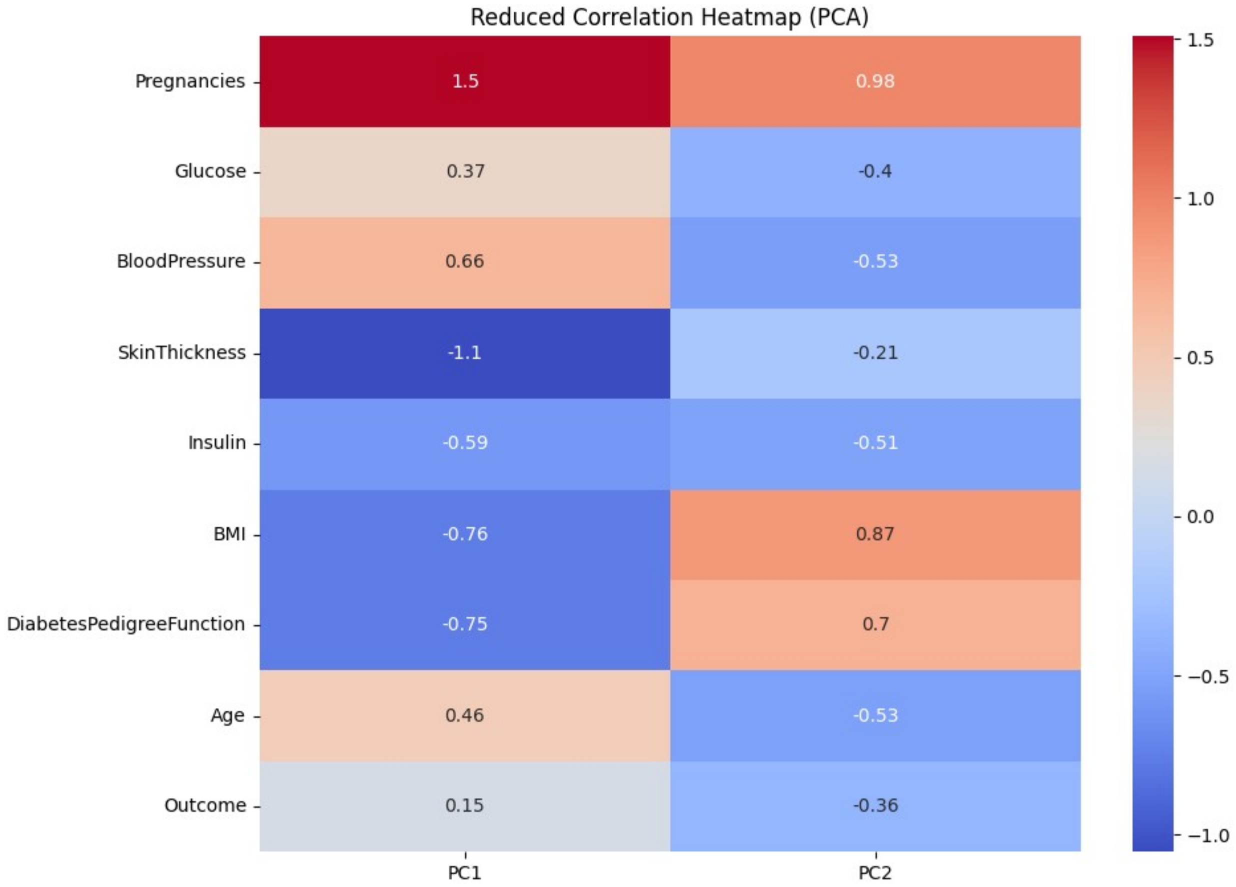
```
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
site-packages/sklearn/metrics/_classification.py:1509:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
site-packages/sklearn/metrics/_classification.py:1509:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
site-packages/sklearn/metrics/_classification.py:1509:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

```
cm = confusion_matrix(y_test, y_pred, labels=knn.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=knn.classes_)
```

```python
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.show()
```

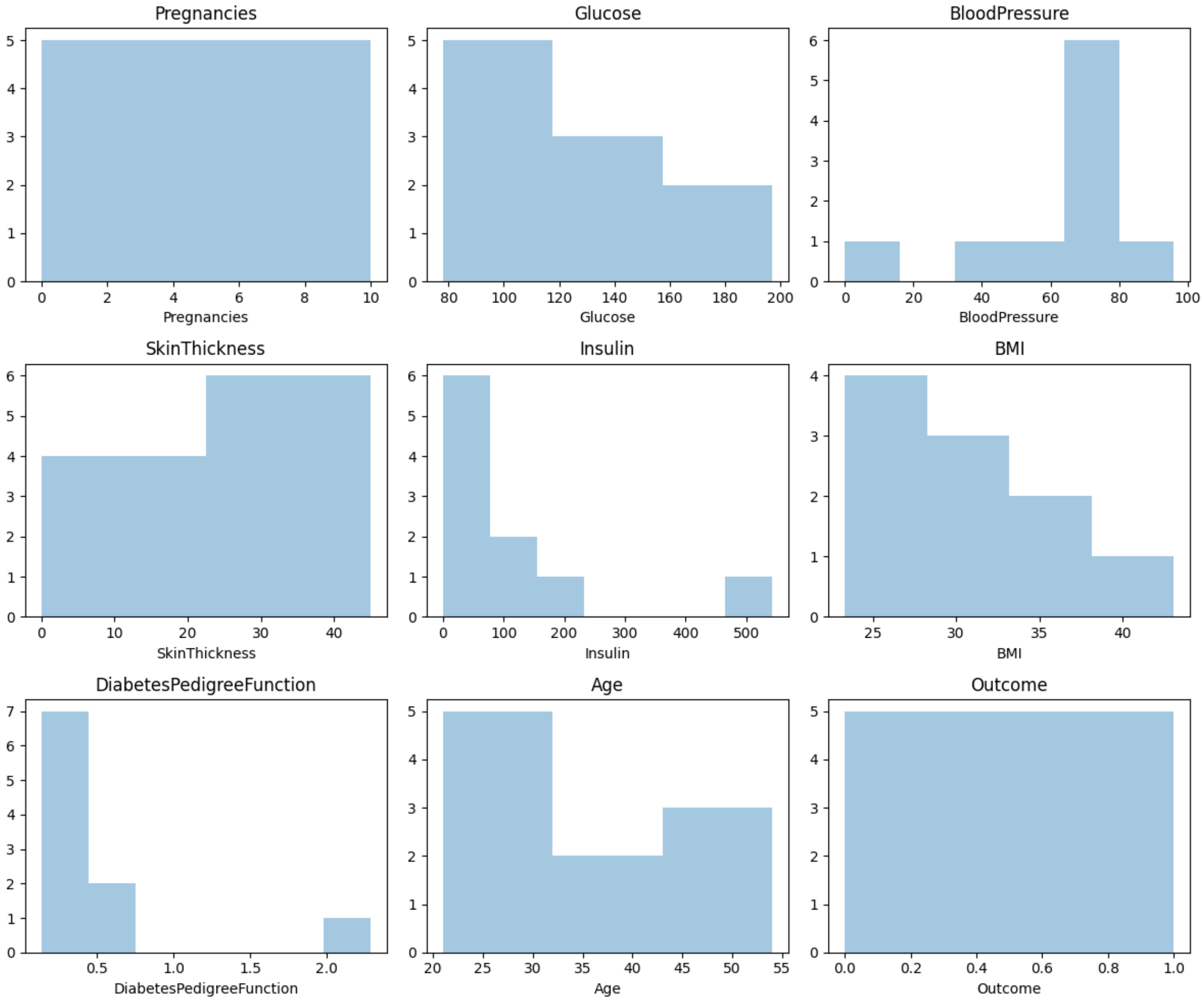


```python
def plot_decision_boundary(X, y, model, title):
    # Define the step size and grid limits
    h = .02  # step size in the mesh
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min,
y_max, h))


pca = PCA(n_components=2)  # Reducing to 2 dimensions
corr_matrix = df.corr().values
pca_corr = pca.fit_transform(corr_matrix)

plt.figure(figsize=(10, 8))
sns.heatmap(pd.DataFrame(pca_corr, index=df.columns, columns=['PC1',
'PC2']), annot=True, cmap='coolwarm')
plt.title('Reduced Correlation Heatmap (PCA)')
plt.show()
```



```
plt.figure(figsize=(12, 10))
for i, col in enumerate(df.columns):
    plt.subplot(3, 3, i+1)
    sns.distplot(df[col], kde=False)
    plt.title(col)
    plt.tight_layout()
plt.show()
```

```
/var/folders/zg/vrs3pzbs675bkpfbq5m4r4yh0000gn/T/
ipykernel_68756/3305820061.py:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[col], kde=False)
```



```
pca = PCA(n_components=2)  # Reducing to 2 dimensions
corr_matrix = df.corr().values
pca_corr = pca.fit_transform(corr_matrix)
```