zinigvvch

March 28, 2025

[1]:
```
# Importing necessary libraries
import pandas as pd                  # For creating and handling DataFrames
import numpy as np                   # For numerical operations and array␣
 ↪handling
import matplotlib as plt             # For data visualization (matplotlib␣
 ↪module)
```

[2]:
```
# Loading the dataset into a DataFrame from a CSV file
df = pd.read_csv("diabetes.csv")   # Reads the CSV file and stores it in the␣
 ↪DataFrame 'df'
```

[3]:
```
df
```

[3]:
```
     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]
```

[4]:

```
df.columns
```

[4]:

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

[5]:

```
df.isnull()
```

[5]:

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin    BMI  \
0          False    False          False          False    False  False
1          False    False          False          False    False  False
2          False    False          False          False    False  False
3          False    False          False          False    False  False
4          False    False          False          False    False  False
..           …      …            …            …      …    …
763        False    False          False          False    False  False
764        False    False          False          False    False  False
765        False    False          False          False    False  False
766        False    False          False          False    False  False
767        False    False          False          False    False  False

     DiabetesPedigreeFunction    Age  Outcome
0                       False  False    False
1                       False  False    False
2                       False  False    False
3                       False  False    False
4                       False  False    False
..                        …    …      …
763                     False  False    False
764                     False  False    False
765                     False  False    False
766                     False  False    False
767                     False  False    False

[768 rows x 9 columns]
```

[3]:

```
# Splitting the dataset into features (X) and target variable (y)

# X → Features (all columns except 'Outcome')
x = df.drop(['Outcome'], axis=1)

# y → Target variable (Outcome column)
y = df['Outcome']
```

[4]:

```
# Importing the train_test_split function from sklearn for data splitting
from sklearn.model_selection import train_test_split

# Splitting the dataset into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(x,y,test_size=0.
 ↪4,random_state=10)
```

[5]:

```
# Importing the Gaussian Naive Bayes classifier from sklearn
from sklearn.naive_bayes import GaussianNB

# Initializing the Gaussian Naive Bayes model
gaussian = GaussianNB()

# Fitting the model to the training data
gaussian.fit(X_train, Y_train)  # Trains the model using the training set
```

[5]:

```
GaussianNB()
```

[6]:

```
# Making predictions on the testing set using the trained Naive Bayes model
Y_pred = gaussian.predict(X_test)  # Predicted labels for the testing set
```

[7]:

```
# Importing performance evaluation metrics from sklearn
from sklearn.metrics import accuracy_score, precision_score, recall_score
```

[8]:

```
# Calculating model performance metrics

# Accuracy: Overall correctness of the model
accuracy = accuracy_score(Y_test, Y_pred)

# Precision: Proportion of true positives among all positive predictions
precision = precision_score(Y_test, Y_pred, average='micro')

# Recall: Proportion of true positives identified correctly
recall = recall_score(Y_test, Y_pred, average='micro')
```

[9]:

```
# Importing performance evaluation metrics from sklearn
from sklearn.metrics import precision_score, confusion_matrix, accuracy_score,␣
 ↪recall_score

# Generating the confusion matrix for the testing set
cm = confusion_matrix(Y_test, Y_pred)  # Compares actual vs predicted labels
```

[20]:

```
cm =confusion_matrix(Y_test,Y_pred)
print("ConfusionMatrix:\n",cm)
```

```
ConfusionMatrix:
 [[166  35]
 [ 47  60]]
```

[21]:

```python
print("Accuracy:", accuracy_score(Y_test, Y_pred))
print("Precision:", precision_score(Y_test, Y_pred, average='weighted'))
print("Recall:", recall_score(Y_test, Y_pred, average='weighted'))
```

```
Accuracy: 0.7337662337662337
Precision: 0.7280092035466387
Recall: 0.7337662337662337
```

[10]:

```python
Y_pred
```

[10]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1,
       1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0],
      dtype=int64)
```

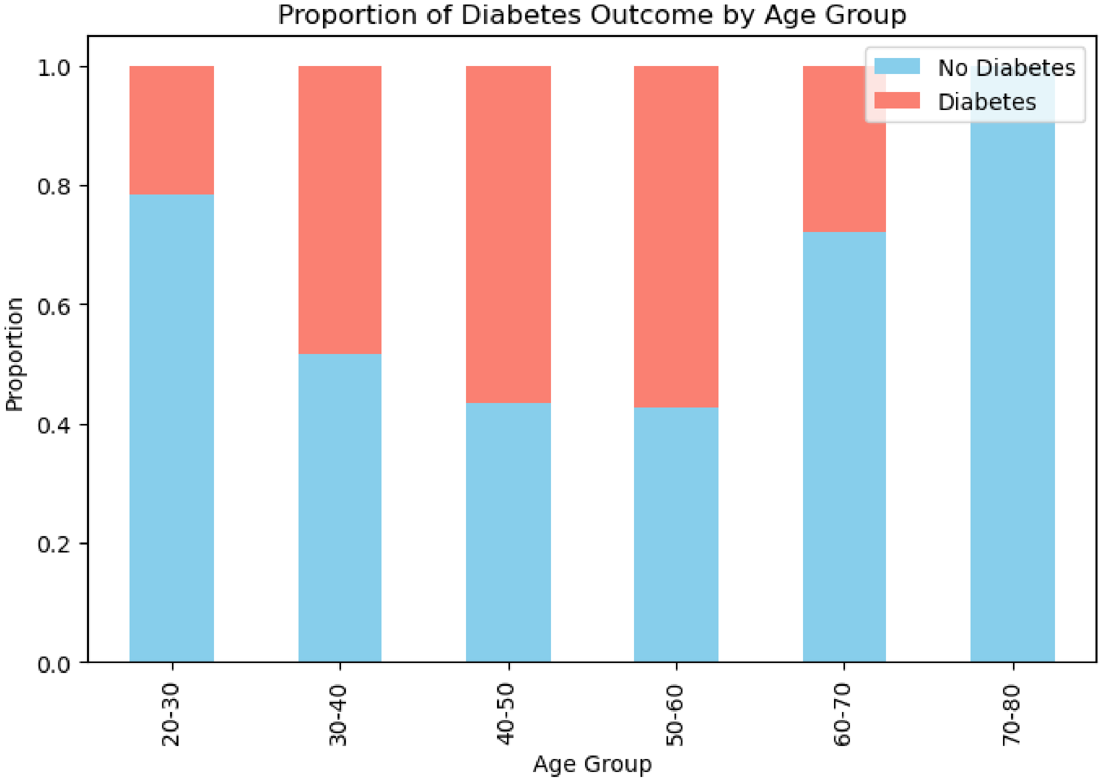
[13]:

```python
import matplotlib.pyplot as plt
# Creating a crosstab to show the proportion of diabetes outcome by age groups
# Binning the 'Age' column into categories for better visualization
df['AgeGroup'] = pd.cut(df['Age'], bins=[20, 30, 40, 50, 60, 70, 80],␣
 ↪labels=['20-30', '30-40', '40-50', '50-60', '60-70', '70-80'])
# Crosstab showing the proportion of diabetes outcome by age group
cm = pd.crosstab(df['AgeGroup'], df['Outcome'], normalize="index")
print("\nCrosstab of Age Group vs Outcome:")
print(cm)

# Plotting the crosstab as a stacked bar chart
cm.plot.bar(figsize=(8, 5), stacked=True, color=['skyblue', 'salmon'])
plt.title("Proportion of Diabetes Outcome by Age Group")
plt.xlabel("Age Group")
plt.ylabel("Proportion")
plt.legend(['No Diabetes', 'Diabetes'])
plt.show()
```

```
Crosstab of Age Group vs Outcome:
```

```
Outcome          0         1
AgeGroup
20-30     0.784173  0.215827
30-40     0.515924  0.484076
40-50     0.433628  0.566372
50-60     0.425926  0.574074
60-70     0.720000  0.280000
70-80     1.000000  0.000000
```



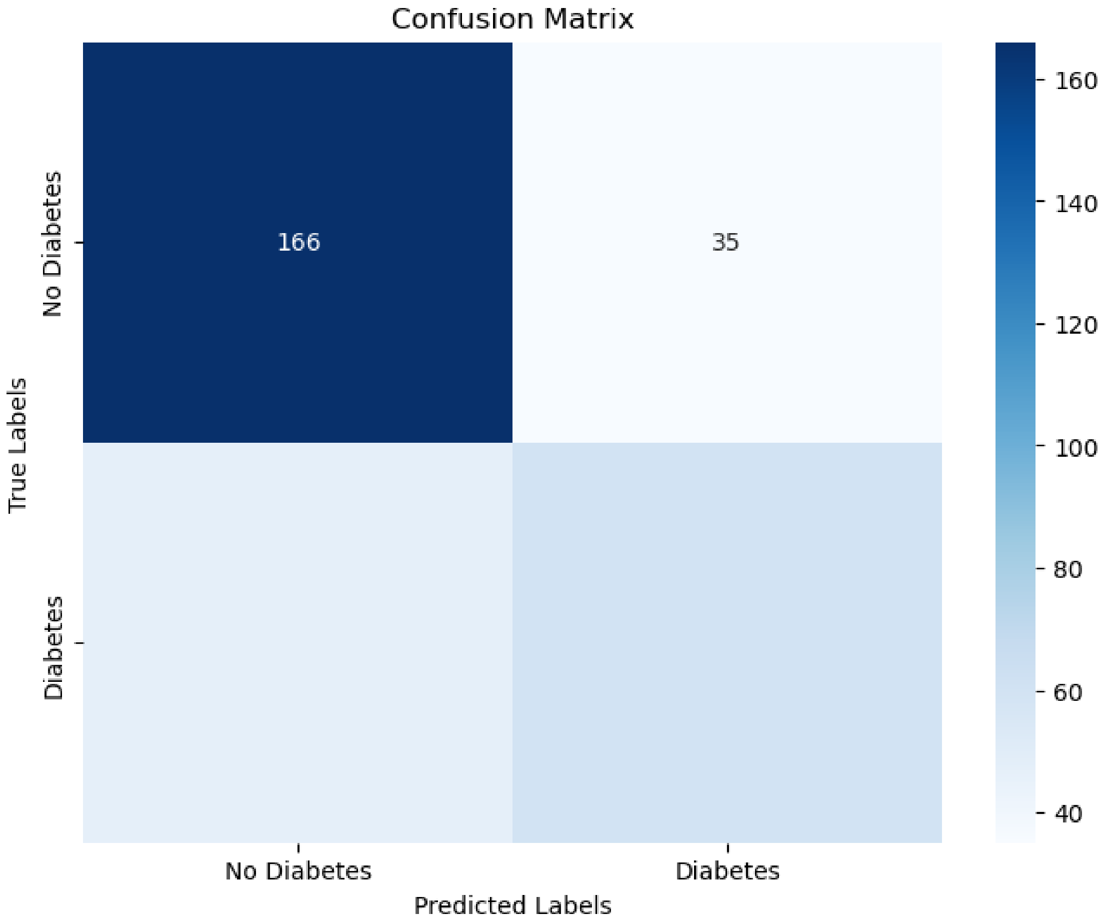
[23]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
```

[24]:
```python
# Generate confusion matrix
cm = confusion_matrix(Y_test, Y_pred)

# Correct class labels
labels = ['No Diabetes', 'Diabetes']  # Use appropriate labels

# Plot confusion matrix
plt.figure(figsize=(8, 6))
```

```
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels,␣
 ↪yticklabels=labels)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```



```
[ ]:
```