sml-ass-naive-bayes-ipynb

April 6, 2023

0.1 Assignment 8: Naive Bayes Classifier

Supervised Machine Learning Lab ### Kaustubh Raykar PRN : 21070126048 AIML A3

Build a Naïve Bayes Classifier to Predict whether income exceeds $50K/yr based on census data. Also known as "Census Income" dataset. Dataset is attached

0.1.1 Importing Libraries

[150]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
```

0.1.2 Upload Dataset

[151]:
```
col=['Age','Worksclass','Fnlwgt','Education','Education_num','Marital_status','Occupation','Re

df = pd.read_csv('/content/Naive bayes dataset.csv', names=col)

df.head(6)
```

[151]:
```
   Age        Worksclass  Fnlwgt  Education  Education_num  \
0   39         State-gov   77516  Bachelors             13
1   50  Self-emp-not-inc   83311  Bachelors             13
2   38           Private  215646    HS-grad              9
3   53           Private  234721       11th              7
4   28           Private  338409  Bachelors             13
5   37           Private  284582    Masters             14

       Marital_status         Occupation   Relationship   Race     Sex  \
0       Never-married       Adm-clerical  Not-in-family  White    Male
1  Married-civ-spouse    Exec-managerial        Husband  White    Male
2            Divorced  Handlers-cleaners  Not-in-family  White    Male
3  Married-civ-spouse  Handlers-cleaners        Husband  Black    Male
4  Married-civ-spouse     Prof-specialty           Wife  Black  Female
```

```
5   Married-civ-spouse     Exec-managerial            Wife    White    Female

   Capital_gain  Capital_loss  Hours_per_week  Native_country  Income
0          2174             0              40   United-States   <=50K
1             0             0              13   United-States   <=50K
2             0             0              40   United-States   <=50K
3             0             0              40   United-States   <=50K
4             0             0              40            Cuba   <=50K
5             0             0              40   United-States   <=50K
```

0.1.3 Data Description

[152]:
```python
print(df.columns)
```

```
Index(['Age', 'Worksclass', 'Fnlwgt', 'Education', 'Education_num',
       'Marital_status', 'Occupation', 'Relationship', 'Race', 'Sex',
       'Capital_gain', 'Capital_loss', 'Hours_per_week', 'Native_country',
       'Income'],
      dtype='object')
```

[153]:
```python
df.dtypes
```

[153]:
```
Age                int64
Worksclass        object
Fnlwgt             int64
Education         object
Education_num      int64
Marital_status    object
Occupation        object
Relationship      object
Race              object
Sex               object
Capital_gain       int64
Capital_loss       int64
Hours_per_week     int64
Native_country    object
Income            object
dtype: object
```

[154]:
```python
df = df.drop(['Fnlwgt'], axis=1)
```

[155]:
```python
df.drop(['Capital_gain', 'Capital_loss', 'Hours_per_week'], axis=1,␣
 ↪inplace=True)
```

[156]:
```python
df
```

[156]:

```
         Age         Worksclass    Education  Education_num  \
0         39          State-gov    Bachelors             13
1         50   Self-emp-not-inc    Bachelors             13
2         38            Private      HS-grad              9
3         53            Private         11th              7
4         28            Private    Bachelors             13
…      …                    …          …              …
32556     27            Private   Assoc-acdm             12
32557     40            Private      HS-grad              9
32558     58            Private      HS-grad              9
32559     22            Private      HS-grad              9
32560     52       Self-emp-inc      HS-grad              9

             Marital_status          Occupation    Relationship    Race  \
0             Never-married        Adm-clerical   Not-in-family   White
1        Married-civ-spouse     Exec-managerial         Husband   White
2                  Divorced   Handlers-cleaners   Not-in-family   White
3        Married-civ-spouse   Handlers-cleaners         Husband   Black
4        Married-civ-spouse      Prof-specialty            Wife   Black
…                       …                   …              …      …
32556    Married-civ-spouse        Tech-support            Wife   White
32557    Married-civ-spouse   Machine-op-inspct         Husband   White
32558               Widowed        Adm-clerical       Unmarried   White
32559         Never-married        Adm-clerical       Own-child   White
32560    Married-civ-spouse     Exec-managerial            Wife   White

            Sex  Native_country  Income
0          Male   United-States   <=50K
1          Male   United-States   <=50K
2          Male   United-States   <=50K
3          Male   United-States   <=50K
4        Female            Cuba   <=50K
…           …              …      …
32556    Female   United-States   <=50K
32557      Male   United-States    >50K
32558    Female   United-States   <=50K
32559      Male   United-States   <=50K
32560    Female   United-States    >50K

[32561 rows x 11 columns]
```

0.1.4 Data Visualisation , EDA

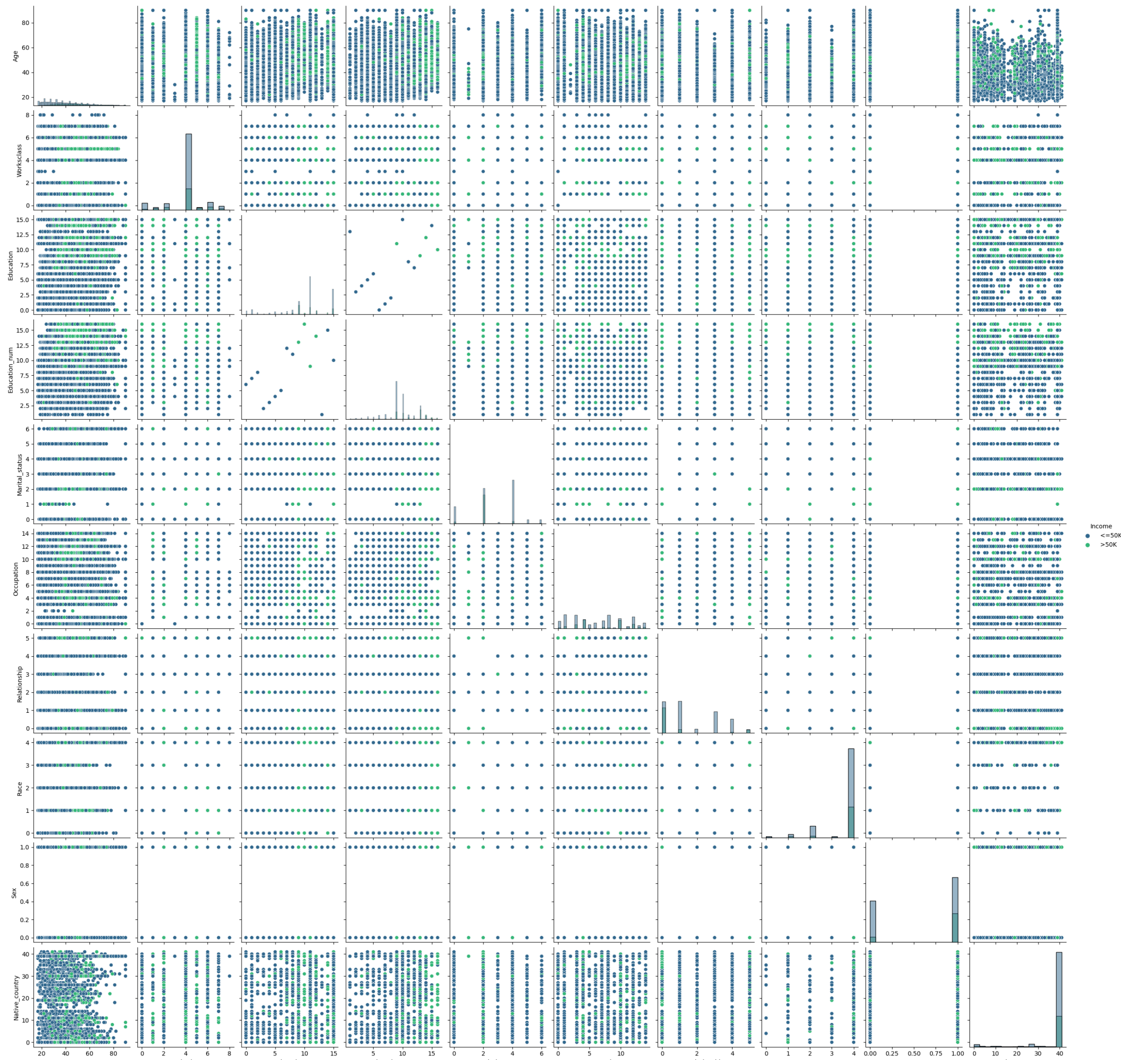
[177]:

```
import seaborn as sns

cols = ['Age', 'Worksclass', 'Education', 'Education_num', 'Marital_status',
 ↪'Occupation', 'Relationship', 'Race', 'Sex', 'Native_country', 'Income']
```

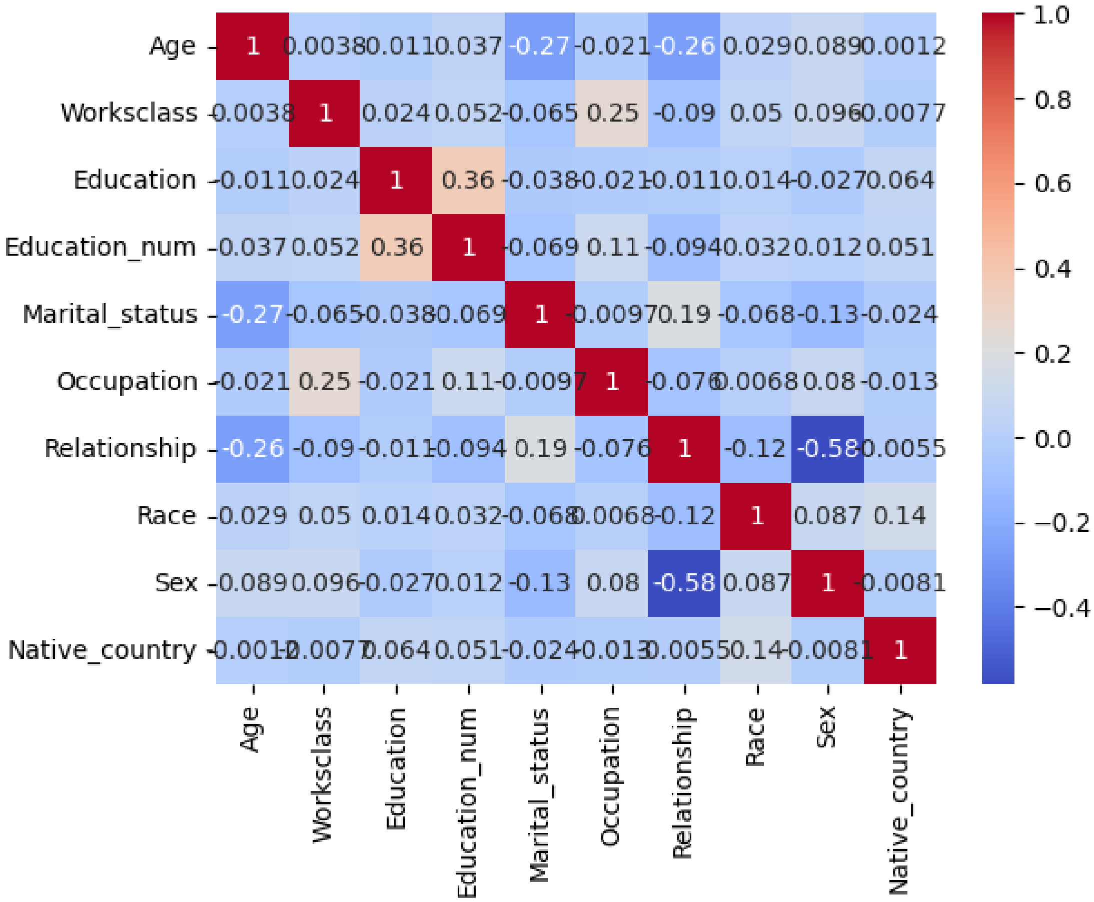
```
sns.pairplot(df[cols], hue='Income', diag_kind='hist', palette='viridis')
```

[177]: <seaborn.axisgrid.PairGrid at 0x7fe464282fa0>



[178]:
```
sns.heatmap(df.corr(), cmap='coolwarm', annot=True)
```

[178]: <Axes: >



[179]:
```python
import plotly.express as px

fig = px.scatter(df, x="Age", y="Income", color="Sex")
fig.show()
```

[181]:
```python
fig = px.histogram(df, x="Age", color="Income")
fig.show()
```

[182]:
```python
fig = px.violin(df, x="Income", y="Education_num", color="Income")
fig.show()
```

[183]:
```python
print(df['Worksclass'].unique())
print(df['Education'].unique())
print(df['Marital_status'].unique())
print(df['Occupation'].unique())
print(df['Relationship'].unique())
print(df['Race'].unique())
print(df['Sex'].unique())
```

```
print(df['Native_country'].unique())
```

```
[7 6 4 1 2 0 5 8 3]
[ 9 11  1 12  6 15  7  8  5 10 14  4  0  3 13  2]
[4 2 0 3 5 1 6]
[ 1  4  6 10  8 12  3 14  5  7 13  0 11  2  9]
[1 0 5 3 4 2]
[4 2 1 0 3]
[1 0]
[39  5 23 19  0 26 35 33 16  9  2 11 20 30 22 31  4  1 37  7 25 36 14 32
  6  8 10 13  3 24 41 29 28 34 38 12 27 40 17 21 18 15]
```

0.1.5 Encoding columns

[184]:
```
# Select the categorical columns to be encoded
cat_cols = ['Worksclass', 'Education', 'Marital_status', 'Occupation',␣
 ↪'Relationship', 'Race', 'Sex', 'Native_country']
# Create an instance of LabelEncoder for each column and fit it to the data
label_encoders = {}
for col in cat_cols:
    label_encoders[col] = LabelEncoder()
    df[col] = label_encoders[col].fit_transform(df[col])

# Print the first 5 rows of the encoded data
print(df.head())
```

```
   Age  Worksclass  Education  Education_num  Marital_status  Occupation  \
0   39           7          9             13               4           1
1   50           6          9             13               2           4
2   38           4         11              9               0           6
3   53           4          1              7               2           6
4   28           4          9             13               2          10

   Relationship  Race  Sex  Native_country  Income
0             1     4    1              39   <=50K
1             0     4    1              39   <=50K
2             1     4    1              39   <=50K
3             0     2    1              39   <=50K
4             5     2    0               5   <=50K
```

[185]:
```
for col in cat_cols:
    le = label_encoders[col]
    print(f"Column: {col}")
    for i, class_label in enumerate(le.classes_):
        print(f"Label {i}: {class_label}")
    print("\n")
```

```
Column: Worksclass
```

```
Label 0: 0
Label 1: 1
Label 2: 2
Label 3: 3
Label 4: 4
Label 5: 5
Label 6: 6
Label 7: 7
Label 8: 8


Column: Education
Label 0: 0
Label 1: 1
Label 2: 2
Label 3: 3
Label 4: 4
Label 5: 5
Label 6: 6
Label 7: 7
Label 8: 8
Label 9: 9
Label 10: 10
Label 11: 11
Label 12: 12
Label 13: 13
Label 14: 14
Label 15: 15


Column: Marital_status
Label 0: 0
Label 1: 1
Label 2: 2
Label 3: 3
Label 4: 4
Label 5: 5
Label 6: 6


Column: Occupation
Label 0: 0
Label 1: 1
Label 2: 2
Label 3: 3
Label 4: 4
Label 5: 5
Label 6: 6
```

```
Label 7: 7
Label 8: 8
Label 9: 9
Label 10: 10
Label 11: 11
Label 12: 12
Label 13: 13
Label 14: 14


Column: Relationship
Label 0: 0
Label 1: 1
Label 2: 2
Label 3: 3
Label 4: 4
Label 5: 5


Column: Race
Label 0: 0
Label 1: 1
Label 2: 2
Label 3: 3
Label 4: 4


Column: Sex
Label 0: 0
Label 1: 1


Column: Native_country
Label 0: 0
Label 1: 1
Label 2: 2
Label 3: 3
Label 4: 4
Label 5: 5
Label 6: 6
Label 7: 7
Label 8: 8
Label 9: 9
Label 10: 10
Label 11: 11
Label 12: 12
Label 13: 13
Label 14: 14
```

```
Label 15: 15
Label 16: 16
Label 17: 17
Label 18: 18
Label 19: 19
Label 20: 20
Label 21: 21
Label 22: 22
Label 23: 23
Label 24: 24
Label 25: 25
Label 26: 26
Label 27: 27
Label 28: 28
Label 29: 29
Label 30: 30
Label 31: 31
Label 32: 32
Label 33: 33
Label 34: 34
Label 35: 35
Label 36: 36
Label 37: 37
Label 38: 38
Label 39: 39
Label 40: 40
Label 41: 41
```

[186]:
```python
# count the number of null values in each column
null_counts = df.isnull().sum()

# print the null counts
print(null_counts)
```

```
Age               0
Worksclass        0
Education         0
Education_num     0
Marital_status    0
Occupation        0
Relationship      0
Race              0
Sex               0
Native_country    0
Income            0
dtype: int64
```

[187]:
```
df
```

[187]:
```
         Age  Worksclass  Education  Education_num  Marital_status  Occupation  \
0         39           7          9             13               4           1
1         50           6          9             13               2           4
2         38           4         11              9               0           6
3         53           4          1              7               2           6
4         28           4          9             13               2          10
…      …           …          …              …               …           …
32556     27           4          7             12               2          13
32557     40           4         11              9               2           7
32558     58           4         11              9               6           1
32559     22           4         11              9               4           1
32560     52           5         11              9               2           4

       Relationship  Race  Sex  Native_country  Income
0                 1     4    1              39   <=50K
1                 0     4    1              39   <=50K
2                 1     4    1              39   <=50K
3                 0     2    1              39   <=50K
4                 5     2    0               5   <=50K
…              …   …   …              …     …
32556             5     4    0              39   <=50K
32557             0     4    1              39    >50K
32558             4     4    0              39   <=50K
32559             3     4    1              39   <=50K
32560             5     4    0              39    >50K

[32561 rows x 11 columns]
```

0.1.6 Splitting into x and y

[188]:
```python
x = df.drop('Income', axis=1)
y = df['Income']
```

[189]:
```
x
```

[189]:
```
         Age  Worksclass  Education  Education_num  Marital_status  Occupation  \
0         39           7          9             13               4           1
1         50           6          9             13               2           4
2         38           4         11              9               0           6
3         53           4          1              7               2           6
4         28           4          9             13               2          10
…      …           …          …              …               …           …
32556     27           4          7             12               2          13
32557     40           4         11              9               2           7
32558     58           4         11              9               6           1
```

```
32559   22          4         11              9               4            1
32560   52          5         11              9               2            4

       Relationship  Race  Sex  Native_country
0                 1     4    1              39
1                 0     4    1              39
2                 1     4    1              39
3                 0     2    1              39
4                 5     2    0               5
…               …   …   …            …
32556             5     4    0              39
32557             0     4    1              39
32558             4     4    0              39
32559             3     4    1              39
32560             5     4    0              39

[32561 rows x 10 columns]
```

[190]:
```
y
```

[190]:
```
0         <=50K
1         <=50K
2         <=50K
3         <=50K
4         <=50K
          …
32556     <=50K
32557      >50K
32558     <=50K
32559     <=50K
32560      >50K
Name: Income, Length: 32561, dtype: object
```

0.1.7 Splitting the data into training and testing sets

[191]:
```
# Split data into train and test sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
 random_state=42)
```

[192]:
```
# Print the size of the train and test sets
print('Size of x_train:', x_train.shape)
print('Size of x_test:', x_test.shape)
print('Size of y_train:', y_train.shape)
print('Size of y_test:', y_test.shape)
```

```
Size of x_train: (26048, 10)
Size of x_test: (6513, 10)
```

```
Size of y_train: (26048,)
Size of y_test: (6513,)
```

0.1.8 Import and apply Naive Bayes model

[193]:
```python
from sklearn.naive_bayes import GaussianNB
classifier= GaussianNB()
classifier.fit(x_train, y_train)
```

[193]:
```
GaussianNB()
```

[194]:
```python
y_pred = classifier.predict(x_test)
```

0.1.9 Classification Results of our Model

[195]:
```python
# Making the Confusion Matrix
from sklearn.metrics import classification_report, confusion_matrix

# Compute the confusion matrix and classification report
cm = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred)

# Print the results
print("Confusion Matrix:\n", cm)
print("\nClassification Report:\n", report)
```

```
Confusion Matrix:
 [[3988  954]
 [ 469 1102]]

Classification Report:
               precision    recall  f1-score   support

       <=50K       0.89      0.81      0.85      4942
        >50K       0.54      0.70      0.61      1571

    accuracy                           0.78      6513
   macro avg       0.72      0.75      0.73      6513
weighted avg       0.81      0.78      0.79      6513
```

[196]:
```python
# Format the output of ac and cm
output = 'The accuracy is {:.2f}%\n\nThe confusion matrix is:\n{}'
output = output.format(ac*100, cm)

# Print the output
print(output)
```

```
The accuracy is 78.15%

The confusion matrix is:
[[3988  954]
 [ 469 1102]]
```

0.1.10 Confusion Matrix graph

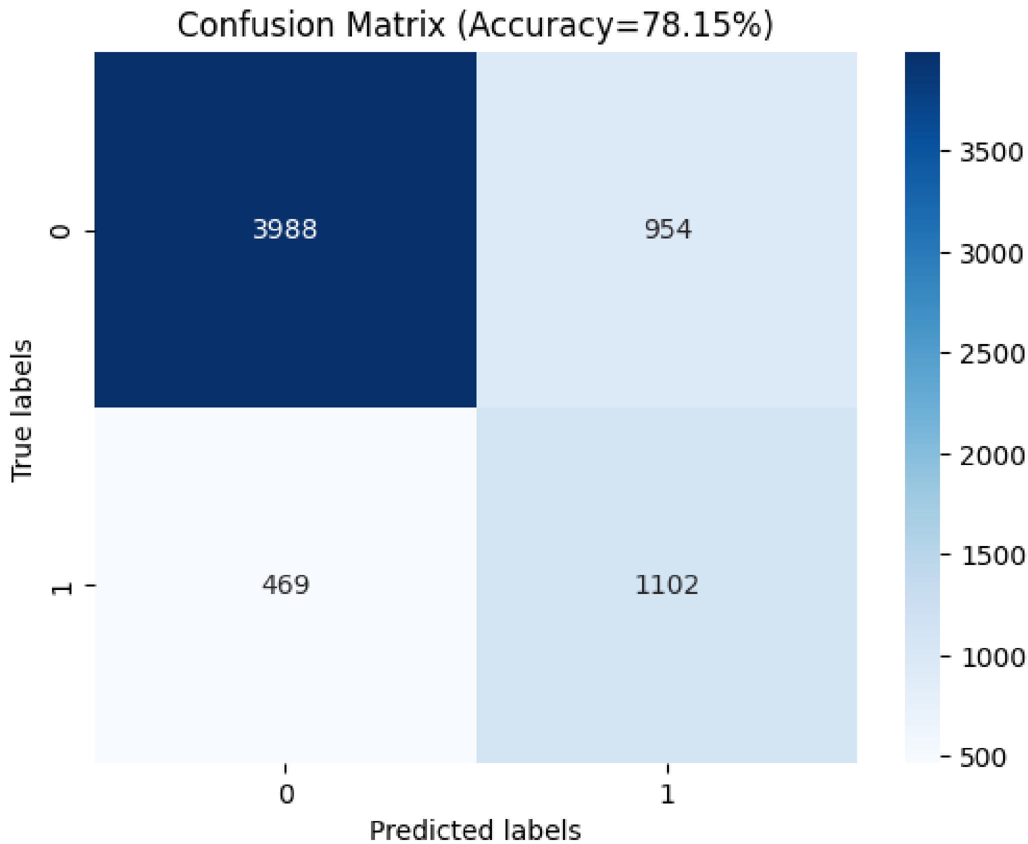
[197]:
```
import seaborn as sns

# Calculate the confusion matrix and accuracy score
ac = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix using seaborn
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

# Add labels and title to the plot
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix (Accuracy={:.2f}%)'.format(ac*100))

# Show the plot
plt.show()
```



0.1.11 Thank you

[197]: