Naive Bayes classifier Algorithm

Importing libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Loading dataset i.e. Adult dataset

```
df = pd.read_csv('adult.csv')
df.head()
```

	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	21
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	
						Married-					

```
df.shape
```

```
    (32560, 15)
```

Rename column names

```
c_n = ['age', 'workclass', 'fnlwgt', 'education', 'education_num', 'marital_status', 'occu
             'race', 'sex', 'capital_gain', 'capital_loss', 'hours_per_week', 'native_coun
```

```
df.columns = c_n
```

```
df.columns
```

```
    Index(['age', 'workclass', 'fnlwgt', 'education', 'education_num',
           'marital_status', 'occupation', 'relationship', 'race', 'sex',
           'capital_gain', 'capital_loss', 'hours_per_week', 'native_country',
           'income'],
          dtype='object')
```

```
df.head()
```

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	rel
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	N
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	

Declaring feature vector and target variable

```
X = df.drop(['income'], axis=1)

y = df['income']
```

Split X and y into training and testing sets

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state =
```

```
X_train.shape, X_test.shape
```

```
((22792, 14), (9768, 14))
```

```
pip install --upgrade category_encoders
```

```
Collecting category_encoders
  Downloading category_encoders-2.4.0-py2.py3-none-any.whl (86 kB)
     |████████████████████████████████| 86 kB 2.5 MB/s
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: pandas>=0.21.1 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dis
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from pa
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages
Installing collected packages: category-encoders
Successfully installed category-encoders-2.4.0
```

```
import category_encoders as ce
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWarnir
  import pandas.util.testing as tm
```

Encoding remaining variables with one-hot encoding

```
the_encoder = ce.OneHotEncoder(cols=['workclass', 'education', 'marital_status', 'occupati
                                 'race', 'sex', 'native_country'])

X_train = the_encoder.fit_transform(X_train)
X_test = the_encoder.transform(X_test)
```

```
X_train.head()
```

	age	workclass_1	workclass_2	workclass_3	workclass_4	workclass_5	workcla
20721	32	1	0	0	0	0	
32097	45	0	1	0	0	0	
25205	47	0	0	1	0	0	
23491	37	0	1	0	0	0	
12367	24	0	1	0	0	0	

5 rows × 108 columns

```
X_test.head()
```

	age	workclass_1	workclass_2	workclass_3	workclass_4	workclass_5	workcla
22278	40	1	0	0	0	0	
8950	46	0	1	0	0	0	
7838	33	0	0	0	0	0	
16505	21	0	1	0	0	0	
19140	59	0	1	0	0	0	

5 rows × 108 columns

Feature scaling

```
the_cols = X_train.columns
```

```
from sklearn.preprocessing import RobustScaler

scaler = RobustScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)
```

```
X_train = pd.DataFrame(X_train, columns=[the_cols])
```

```
X_test = pd.DataFrame(X_test, columns=[the_cols])
```

```
X_train.head()
```

	age	workclass_1	workclass_2	workclass_3	workclass_4	workclass_5	workclass_
0	-0.25	1.0	-1.0	0.0	0.0	0.0	0.
1	0.40	0.0	0.0	0.0	0.0	0.0	0.
2	0.50	0.0	-1.0	1.0	0.0	0.0	0.
3	0.00	0.0	0.0	0.0	0.0	0.0	0.
4	-0.65	0.0	0.0	0.0	0.0	0.0	0.

5 rows × 108 columns

Model training

- Train a Gaussian Naive Bayes classifier on the training set
- instantiate the model
- fit the model

```
from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB()

gnb.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarni
  FutureWarning,
GaussianNB()
```

Predicting results

```
y_pred = gnb.predict(X_test)

y_pred
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarnir
  FutureWarning,
array([' >50K', ' <=50K', ' <=50K', ..., ' >50K', ' <=50K', ' <=50K'],
      dtype='<U6')
```

Checking model accuracy score

```
from sklearn.metrics import accuracy_score

print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score: 0.8062
```

```
print('Training set score: {:.4f}'.format(gnb.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(gnb.score(X_test, y_test)))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarnir
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarnir
  FutureWarning,
Training set score: 0.8021
Test set score: 0.8062
```

Checking class distribution in test set

```
y_test.value_counts()
```

```
 <=50K    7454
 >50K     2314
Name: income, dtype: int64
```

Printing the Confusion Matrix and slicing it into four pieces

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

print('Confusion matrix\n\n', cm)
print('\nTrue Positives(TP) = ', cm[0,0])
print('\nTrue Negatives(TN) = ', cm[1,1])
print('\nFalse Positives(FP) = ', cm[0,1])
print('\nFalse Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix

 [[5970 1484]
 [ 409 1905]]
```

```
     True Positives(TP) =  5970

     True Negatives(TN) =  1905

     False Positives(FP) =  1484

     False Negatives(FN) =  409
```

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
                   precision    recall  f1-score   support

            <=50K       0.94      0.80      0.86      7454
             >50K       0.56      0.82      0.67      2314

         accuracy                           0.81      9768
        macro avg       0.75      0.81      0.77      9768
     weighted avg       0.85      0.81      0.82      9768
```

```
TP = cm[0,0]
TN = cm[1,1]
FP = cm[0,1]
FN = cm[1,0]
```

```
classification_accuracy = (TP + TN) / float(TP + TN + FP + FN)

print('Classification accuracy : {0:0.4f}'.format(classification_accuracy))
```

```
     Classification accuracy : 0.8062
```

```
classification_error = (FP + FN) / float(TP + TN + FP + FN)

print('Classification error : {0:0.4f}'.format(classification_error))
```

```
     Classification error : 0.1938
```

```
precision = TP / float(TP + FP)

print('Precision : {0:0.4f}'.format(precision))
```

```
     Precision : 0.8009
```