I decided to develop a classification model for 'cses4_cut.csv' data. My input variables are 'D2003' (EDUCATION), 'D2010' (CURRENT EMPLOYMENT STATUS), 'age'. My target variable is ‘voted’.

First, I replaced missing values in 'D2003', 'D2010' columns with most frequent items in each column. Then I constructed features and target matrices. In features matrix, 'D2003', 'D2010' columns have categorical information. Therefore, I used one-hot encoding on features matrix.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer

#read from csv file
df = pd.read_csv('cses4_cut.csv')
#construct features matrix, my features are D2003: education and age
X = df[['D2003', 'D2010', 'age']]
#target matrix
Y = df['voted']

#for D2003 and D2010 columns replaced missing values with most_frequent items
X_processed = SimpleImputer(strategy='most_frequent', missing_values=99)
X_processed = X_processed.fit(X[['D2003']])
X['D2003'] = X_processed.transform(X[['D2003']])
X_processed = X_processed.fit(X[['D2010']])
X['D2010'] = X_processed.transform(X[['D2010']])

# education is categorical, thefore I will apply one-hot encoding
# creating instance of one-hot-encoder
X_encoded = OneHotEncoder(sparse=True, handle_unknown='ignore')
# passing bridge-types-cat column
X_encoded = pd.DataFrame(X_encoded.fit_transform(X[['D2003', 'D2010']]).toarray())
# merge encoded columns with age column
X_encoded = X_encoded.join(X['age'])
```

Then, I decided to use Gaussian naive Bayes model as classifier. I split existing data as train (default %75) data and test (default %25) data. I fitted the model, predicted values for Xtest, calculated accuracy of model. Accuracy score was appr. 0.79.

```
#split the data as train and test
Xtrain, Xtest, Ytrain, Ytest = train_test_split(X_encoded, Y,
random_state=1)
```

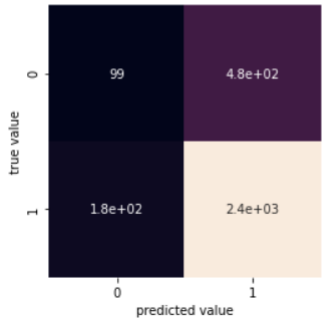
```
#First, I will use Gaussian naive Bayes model for classification
model = GaussianNB() #used Gaussian naive Bayes model for classification
model.fit(Xtrain, Ytrain) #fit the model
Y_model = model.predict(Xtest) #predict y values for x test

accuracy_score(Ytest, Y_model) #calculates accuracy score of mode
```

```
0.7895920334082879
```

I plotted confusion matrix to see model’s output.

```
#where did we go wrong? Confusion matrix shows frequency of misclassification
import seaborn as sns
from sklearn.metrics import confusion_matrix
mat = confusion_matrix(Ytest, Y_model)
sns.heatmap(mat, square=True, annot=True, cbar=False)
plt.xlabel('predicted value')
plt.ylabel('true value')
```

```
Text(91.68, 0.5, 'true value')
```



Then, I decided to use KNeighborsClassifier as classifier. I fitted the model, predicted values for Xtest, calculated accuracy of model. Accuracy score was appr. 0.77.

```
#Second, I will use KNeighborsClassifier model for classification

model2 = KNeighborsClassifier(n_neighbors=1)
model2.fit(Xtrain, Ytrain)

Y_KN_model = model2.predict(Xtest)
accuracy_score(Ytest, Y_KN_model)
```

```
0.7654995181496949
```

I decided to optimize KNeighborsClassifier model because it has high accuracy score. For optimization, I used GridSearchCV module. By help of this module, I tried following hyperparameters:

- **'n_neighbors'**: np.arange(20),
- **'weights'**: ['uniform', 'distance'],
- **'algorithm'** : ['auto', 'ball_tree', 'kd_tree', 'brute']

GridSearchCV module returned best parameters as following:

- **'algorithm'**: 'brute',
- **'n_neighbors'**: 17,
- **'weights'**: 'uniform'

```
#For model optimization, I will use GridSearchCV

from sklearn.model_selection import GridSearchCV
import numpy as np

grid_params = {'n_neighbors': np.arange(20),
               'weights': ['uniform', 'distance'],
               'algorithm' : ['auto', 'ball_tree', 'kd_tree', 'brute']}

grid = GridSearchCV(KNeighborsClassifier(), grid_params, verbose = 1, cv = 7, n_jobs = -1)

grid.fit(Xtrain,Ytrain)
grid.best_params_
```

```
Fitting 7 folds for each of 160 candidates, totalling 1120 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    2.1s
[Parallel(n_jobs=-1)]: Done 560 tasks      | elapsed:    6.3s
[Parallel(n_jobs=-1)]: Done 1120 out of 1120 | elapsed:   20.2s finished
```

```
{'algorithm': 'brute', 'n_neighbors': 17, 'weights': 'uniform'}
```

Finally, I run KNeighborsClassifier with best hyperparameter values. I got better accuracy score (appr. 0.86).

```
#Finally, I will use KNeighborsClassifier model with best parameters

model = KNeighborsClassifier(n_neighbors=17, weights='uniform', algorithm='brute')
model.fit(Xtrain, Ytrain)

Y_KN_model2 = model.predict(Xtest)
accuracy_score(Ytest, Y_KN_model2)
```

```
0.8445229681978799
```

Then I plotted confusion matrix to see model's output.

```
#where did we go wrong? Confusion matrix shows frequency of misclassification
mat = confusion_matrix(Ytest, Y_KN_model2)
sns.heatmap(mat, square=True, annot=True, cbar=False)
plt.xlabel('predicted value')
plt.ylabel('true value')
```

```
Text(91.68, 0.5, 'true value')
```