I decided to develop a classification model for 'cses4_cut.csv' data. First, I constructed features and target matrices. In features matrix, all columns except ‘age’ 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

#read from csv file
df = pd.read_csv('cses4_cut.csv')
#construct features matrix, my features are D2003: education and age
X = df.iloc[:,1:32]
#target matrix
Y = df['voted']

# 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.iloc[:,1:31]).toarray())
# merge with main df bridge_df on key values
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 low (appr. 0.32).

```
#First, I will use Gaussian naive Bayes model for classification

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

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
```

```
0.31769996787664634
```

Then, I decided to use KNeighborsClassifier 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 high (appr. 0.8).

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

X_KNtrain, X_KNtest, Y_KNtrain, Y_KNtest = train_test_split(X_encoded, Y,
random_state=1)
model2 = KNeighborsClassifier(n_neighbors=1)
model2.fit(X_KNtrain, Y_KNtrain)

Y_KN_model = model2.predict(X_KNtest)
accuracy_score(Y_KNtest, Y_KN_model)
```

```
0.799229039511725
```

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'**: 'auto',
- **'n_neighbors'**: 18,
- **'weights'**: 'distance'

```
#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(X_KNtrain,Y_KNtrain)
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:   11.3s
[Parallel(n_jobs=-1)]: Done 184 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done 434 tasks      | elapsed:  3.8min
[Parallel(n_jobs=-1)]: Done 784 tasks      | elapsed:  7.1min
[Parallel(n_jobs=-1)]: Done 1120 out of 1120 | elapsed:  9.2min finished
```

```
{'algorithm': 'auto', 'n_neighbors': 18, 'weights': 'distance'}
```

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

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

X_KNtrain2, X_KNtest2, Y_KNtrain2, Y_KNtest2 = train_test_split(X_encoded, Y,
random_state=1)
model = KNeighborsClassifier(n_neighbors=18, weights='distance', algorithm='auto')
model.fit(X_KNtrain2, Y_KNtrain2)

Y_KN_model2 = model.predict(X_KNtest2)
accuracy_score(Y_KNtest2, Y_KN_model2)
```

```
0.8586572438162544
```