```
Random forest is a supervised learning algorithm
It has 2 mechanism (variation)
    1) Classfication problems
    2) Regression Problems

It is one of the most flexible and easy to use algroithm.

It creates decision tree on the given data samples, gets prediction from each tree and select the best situation by means of voting

It is also a pretty good indicator of feature importance.

Random forest combines multiple decision tree resulting in a forest of trees hence the name random forest came into picture.

The higher the number of tress in the forest, the more accurate is the result.
```

In [ ]:
```

```

Algorithm

```
RFA can be divided into 2 stages:

1) in the first stage we randomly select "k" features out of total "n" features and build random forest
   where k < n

2) Among the k features calculate the node 'd' using the best split point

3) Split the node ('d') into daughter node using the best split

4) Repeat the step until 'l' number of nodes has been reached

5) build forest by repeating step 1 to 4 for 'x' number of time to create 'x' number of tree
```

In [ ]:
```

```

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
data = pd.read_csv('car_evaluation.csv')
```

In [3]:
```
1 data
```

Out[3]:

	vhigh	vhigh.1	2	2.1	small	low	unacc
0	vhigh	vhigh	2	2	small	med	unacc
1	vhigh	vhigh	2	2	small	high	unacc
2	vhigh	vhigh	2	2	med	low	unacc
3	vhigh	vhigh	2	2	med	med	unacc
4	vhigh	vhigh	2	2	med	high	unacc
...	...	...	...	...	...	...	...
1722	low	low	5more	more	med	med	good
1723	low	low	5more	more	med	high	vgood
1724	low	low	5more	more	big	low	unacc
1725	low	low	5more	more	big	med	good
1726	low	low	5more	more	big	high	vgood

1727 rows × 7 columns

In [4]:
```
1 col_names = ['buying','maintenance','doors','person','luggae','safety','class']
```

In [5]:
```
1 data.columns = col_names
```

In [6]:
```
1 data
```

Out[6]:

	buying	maintenance	doors	person	luggae	safety	class
0	vhigh	vhigh	2	2	small	med	unacc
1	vhigh	vhigh	2	2	small	high	unacc
2	vhigh	vhigh	2	2	med	low	unacc
3	vhigh	vhigh	2	2	med	med	unacc
4	vhigh	vhigh	2	2	med	high	unacc
...	...	...	...	...	...	...	...
1722	low	low	5more	more	med	med	good
1723	low	low	5more	more	med	high	vgood
1724	low	low	5more	more	big	low	unacc
1725	low	low	5more	more	big	med	good
1726	low	low	5more	more	big	high	vgood

1727 rows × 7 columns

In [7]:
```
1 data['person']
```

Out[7]:
```
0          2
1          2
2          2
3          2
4          2
        ...
1722    more
1723    more
1724    more
1725    more
1726    more
Name: person, Length: 1727, dtype: object
```

In [8]:
```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1727 entries, 0 to 1726
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   buying       1727 non-null   object
 1   maintenance  1727 non-null   object
 2   doors        1727 non-null   object
 3   person       1727 non-null   object
 4   luggae       1727 non-null   object
 5   safety       1727 non-null   object
 6   class        1727 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

In [9]:
```
1 data['buying'].value_counts()
```

Out[9]:
```
high     432
low      432
med      432
vhigh    431
Name: buying, dtype: int64
```

In [10]:
```
col_names = ['buying','maintenance','doors','person','luggae','safety','class']
for item in col_names:
    print(data[item].value_counts())
    print("==========================\n")
```

```
high     432
low      432
med      432
vhigh    431
Name: buying, dtype: int64
==========================

high     432
low      432
med      432
vhigh    431
Name: maintenance, dtype: int64
==========================

4        432
5more    432
3        432
2        431
Name: doors, dtype: int64
==========================

4       576
more    576
2       575
Name: person, dtype: int64
==========================

med      576
big      576
small    575
Name: luggae, dtype: int64
==========================

high    576
med     576
low     575
Name: safety, dtype: int64
==========================

unacc    1209
acc       384
good       69
vgood      65
Name: class, dtype: int64
==========================
```

In [11]:
```
1 data['class']
```

Out[11]:
```
0       unacc
1       unacc
2       unacc
3       unacc
4       unacc
        ...
1722     good
1723    vgood
1724    unacc
1725     good
1726    vgood
Name: class, Length: 1727, dtype: object
```

In [12]:
```
1 data.isnull().sum()
```

Out[12]:
```
buying         0
maintenance    0
doors          0
person         0
luggae         0
safety         0
class          0
dtype: int64
```

In [ ]:
```
1
```

In [13]:
```
1 X = data.drop(['class'],axis=1)
```

In [14]:
```
1 X
```

Out[14]:

	buying	maintenance	doors	person	luggae	safety
0	vhigh	vhigh	2	2	small	med
1	vhigh	vhigh	2	2	small	high
2	vhigh	vhigh	2	2	med	low
3	vhigh	vhigh	2	2	med	med
4	vhigh	vhigh	2	2	med	high
...	...	...	...	...	...	...
1722	low	low	5more	more	med	med
1723	low	low	5more	more	med	high
1724	low	low	5more	more	big	low
1725	low	low	5more	more	big	med
1726	low	low	5more	more	big	high

1727 rows × 6 columns

In [15]:
```
1 y = data['class']
```

In [16]:
```
1 y
```

Out[16]:
```
0       unacc
1       unacc
2       unacc
3       unacc
4       unacc
        ...
1722     good
1723    vgood
1724    unacc
1725     good
1726    vgood
Name: class, Length: 1727, dtype: object
```

In [ ]:
```
1
```

In [17]:
```
1 from sklearn.model_selection import train_test_split
2 X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=42)
```

In [18]:
```
1 X_train.shape
```

Out[18]:
```
(1208, 6)
```

In [20]:
```
1 X_test.shape
```

Out[20]:
```
(519, 6)
```

In [ ]:
```
1
```

In [21]:
```
1 y_train.shape
```

Out[21]:
```
(1208,)
```

In [22]:
```
1 y_test.shape
```

Out[22]:
```
(519,)
```

In [ ]:
```
1
```

In [23]:
```
1 X_train.head()
```

Out[23]:

	buying	maintenance	doors	person	luggae	safety
1177	med	med	5more	4	big	high
585	high	high	3	more	small	med
1551	low	med	3	4	med	med
727	high	med	4	more	big	high
707	high	med	4	2	big	low

In [24]:
```
1 X_train.dtypes
```

Out[24]:
```
buying         object
maintenance    object
doors          object
person         object
luggae         object
safety         object
dtype: object
```

In [26]:
```
1 import category_encoders as ce
```

In [27]:
```
1 encoder = ce.OrdinalEncoder(cols=['buying','maintenance','doors','person','luggae','safety'])
2 X_train = encoder.fit_transform(X_train)
3 X_test = encoder.transform(X_test)
```

```
/home/punit/anaconda3/lib/python3.8/site-packages/category_encoders/utils.py:21: FutureWarning: is_categorical is deprecated and will be removed in a future versi
on.  Use is_categorical_dtype instead
  elif pd.api.types.is_categorical(cols):
```

In [28]:
```
1 X_train
```

Out[28]:

	buying	maintenance	doors	person	luggae	safety
1177	1	1	1	1	1	1
585	2	2	2	2	2	2
1551	3	1	2	1	3	2
727	2	1	3	2	1	1
707	2	1	3	3	1	3
...	...	...	...	...	...	...
1130	1	1	2	2	1	3
1294	1	4	1	2	1	1
860	2	4	1	2	1	3
1459	3	2	3	3	2	1
1126	1	1	2	2	2	1

1208 rows × 6 columns

In [29]:
```
1 X_test
```

Out[29]:

	buying	maintenance	doors	person	luggae	safety
599	2	2	3	3	1	3
932	1	3	3	1	1	3
628	2	2	1	3	1	1
1497	3	2	1	1	3	2
1262	1	4	3	2	3	3
...	...	...	...	...	...	...
490	2	3	3	3	3	1
1276	1	4	1	3	1	1
287	4	1	3	2	2	3
701	2	1	3	3	2	3
1713	3	4	1	1	3	2

519 rows × 6 columns

In [30]:
```
1 from sklearn.tree import DecisionTreeClassifier
```

In [31]:
```
1 gini_res = DecisionTreeClassifier(criterion='gini',max_depth=3,random_state=0)
```

In [32]:
```
1 gini_res.fit(X_train,y_train)
```

Out[32]:
```
DecisionTreeClassifier(max_depth=3, random_state=0)
```

In [33]:
```
1 y_pred = gini_res.predict(X_test)
```

In [34]:
```
1 from sklearn.metrics import accuracy_score
2 print('Model accuracy score with gini index is {0:0.4f}'.format(accuracy_score(y_test,y_pred)))
```

```
Model accuracy score with gini index is 0.8150
```

In [ ]:
```
1
```

In [35]:
```
1 y_pred_train_gini = gini_res.predict(X_train)
```

In [36]:
```
1 y_pred_train_gini
```

Out[36]:
```
array(['acc', 'acc', 'acc', ..., 'unacc', 'unacc', 'acc'], dtype=object)
```

In [37]:
```
1 print('Training Data accuracy {0:04f}'.format(accuracy_score(y_train,y_pred_train_gini)))
```

```
Training Data accuracy 0.801325
```

In [ ]:
```
1
```

In [38]:
```
print('Training Score is : {:4f}'.format(gini_res.score(X_train,y_train)))
print('Testing Score is : {:4f}'.format(gini_res.score(X_test,y_test)))
```

```
Training Score is : 0.801325
Testing Score is : 0.815029
```

In [40]:
```
X_train
```

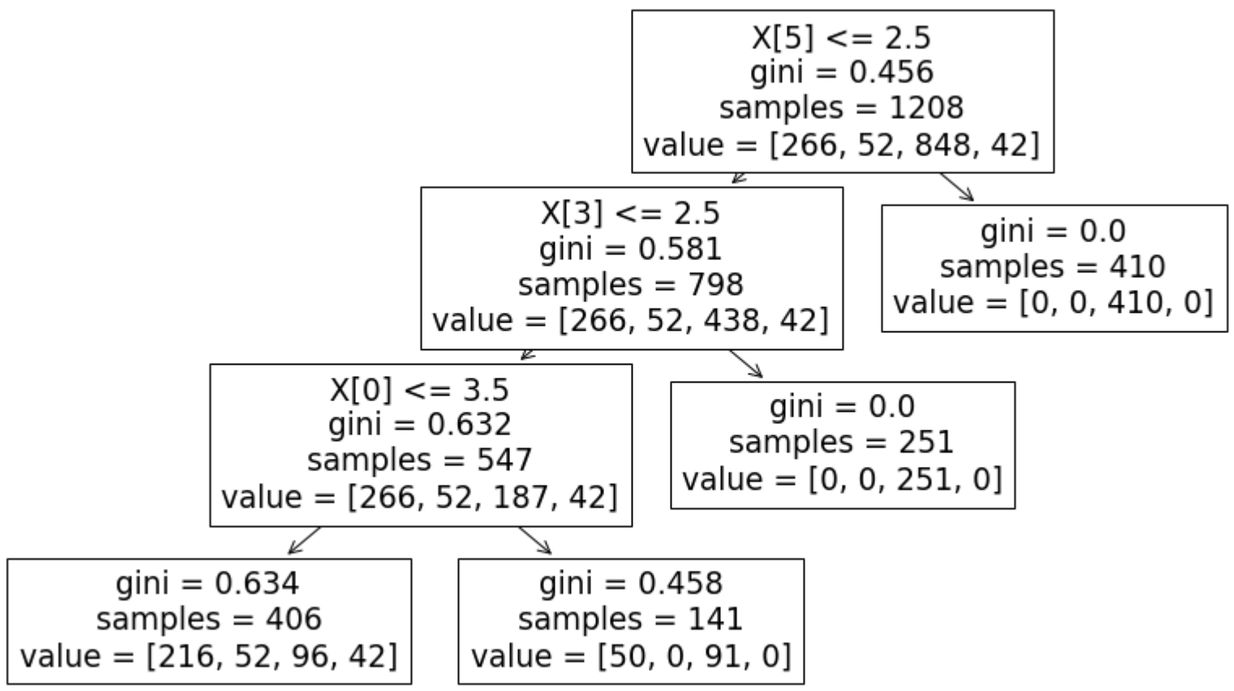
Out[40]:

	buying	maintenance	doors	person	luggae	safety
1177	1	1	1	1	1	1
585	2	2	2	2	2	2
1551	3	1	2	1	3	2
727	2	1	3	2	1	1
707	2	1	3	3	1	3
...	...	...	...	...	...	...
1130	1	1	2	2	1	3
1294	1	4	1	2	1	1
860	2	4	1	2	1	3
1459	3	2	3	3	2	1
1126	1	1	2	2	2	1

1208 rows × 6 columns

In [39]:
```
plt.figure(figsize=(14,8))
from sklearn import tree
tree.plot_tree(gini_res.fit(X_train,y_train))
```

Out[39]:
```
[Text(520.8000000000001, 380.52, 'X[5] <= 2.5\ngini = 0.456\nsamples = 1208\nvalue = [266, 52, 848, 42]'),
 Text(390.6, 271.8, 'X[3] <= 2.5\ngini = 0.581\nsamples = 798\nvalue = [266, 52, 438, 42]'),
 Text(260.40000000000003, 163.07999999999998, 'X[0] <= 3.5\ngini = 0.632\nsamples = 547\nvalue = [266, 52, 187, 42]'),
 Text(130.20000000000002, 54.360000000000014, 'gini = 0.634\nsamples = 406\nvalue = [216, 52, 96, 42]'),
 Text(390.6, 54.360000000000014, 'gini = 0.458\nsamples = 141\nvalue = [50, 0, 91, 0]'),
 Text(520.8000000000001, 163.07999999999998, 'gini = 0.0\nsamples = 251\nvalue = [0, 0, 251, 0]'),
 Text(651.0000000000001, 271.8, 'gini = 0.0\nsamples = 410\nvalue = [0, 0, 410, 0]')]
```



In [41]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

         acc       0.56      0.81      0.67       118
        good       0.00      0.00      0.00        17
       unacc       0.94      0.91      0.92       361
       vgood       0.00      0.00      0.00        23

    accuracy                           0.82       519
   macro avg       0.38      0.43      0.40       519
weighted avg       0.78      0.82      0.79       519


/home/punit/anaconda3/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/punit/anaconda3/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/punit/anaconda3/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:

```

```

In [49]:

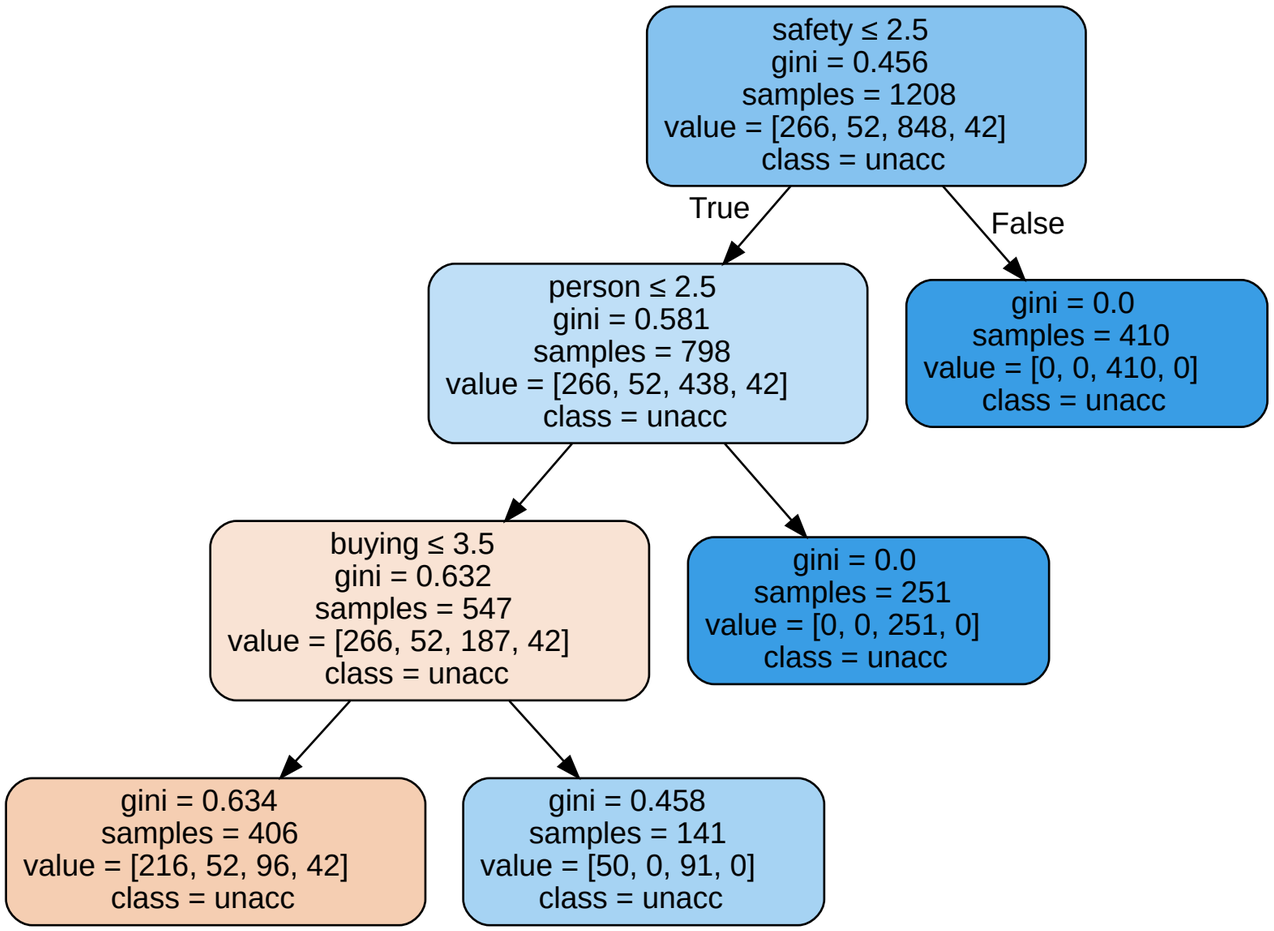
```
import graphviz

tree_data = tree.export_graphviz(gini_res,out_file=None,feature_names=X_train.columns,class_names=y_train,
                                 filled=True,rounded=True,special_characters=True)
```

In [50]:

```
graph = graphviz.Source(tree_data)
```

In [51]:
```
1 graph
```

Out[51]:



In [42]:
```
1 print(y_test,y_pred)
```

```
599     unacc
932     unacc
628     unacc
1497      acc
1262    unacc
        ...
490     unacc
1276    unacc
287     unacc
701     unacc
1713     good
Name: class, Length: 519, dtype: object ['unacc' 'unacc' 'unacc' 'acc' 'unacc' 'acc' 'unacc' 'unacc' 'acc' 'unacc'
 'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'unacc' 'acc'
 'unacc' 'unacc' 'acc' 'unacc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc'
 'unacc' 'acc' 'unacc' 'unacc' 'acc' 'unacc' 'unacc' 'acc' 'unacc' 'acc'
 'acc' 'acc' 'acc' 'unacc' 'unacc' 'acc' 'acc' 'unacc' 'unacc' 'unacc'
 'unacc' 'unacc' 'unacc' 'acc' 'acc' 'acc' 'unacc' 'unacc' 'unacc' 'unacc'
 'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'acc' 'unacc' 'acc'
 'unacc' 'acc' 'acc' 'unacc' 'acc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc'
 'unacc' 'acc' 'unacc' 'unacc' 'acc' 'unacc' 'acc' 'unacc' 'unacc' 'acc'
 'acc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc'
 'unacc' 'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'acc' 'unacc'
 'unacc' 'acc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc'
 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'acc' 'acc' 'unacc' 'acc'
 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc'
 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'acc' 'acc' 'acc'
 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'acc' 'unacc' 'acc' 'unacc' 'unacc'
 'acc' 'unacc' 'unacc' 'unacc' 'acc' 'acc' 'acc' 'acc' 'unacc' 'acc'
 'unacc' 'acc' 'unacc' 'acc' 'acc' 'unacc' 'acc' 'unacc' 'acc' 'acc' 'acc'
 'unacc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc' 'acc' 'acc' 'unacc' 'unacc'
 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc' 'unacc'
 'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc'
 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc'
 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc' 'unacc'
 'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc'
 'unacc' 'unacc' 'acc' 'acc' 'unacc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc'
 'unacc' 'unacc' 'acc' 'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc'
 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'acc' 'unacc' 'unacc'
 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc'
 'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'unacc'
 'unacc' 'acc' 'acc' 'acc' 'unacc' 'acc' 'acc' 'unacc' 'acc' 'unacc' 'acc'
 'unacc' 'acc' 'unacc' 'unacc' 'acc' 'acc' 'unacc' 'unacc' 'unacc' 'unacc'
 'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'unacc'
 'unacc' 'acc' 'acc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'acc'
 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'acc' 'unacc' 'unacc' 'unacc'
 'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc'
 'unacc' 'unacc' 'acc' 'acc' 'unacc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc'
 'unacc' 'unacc' 'acc' 'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc'
 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'acc' 'unacc' 'unacc'
 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc'
 'acc' 'unacc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'unacc'
 'unacc' 'acc' 'acc' 'acc' 'unacc' 'acc' 'acc' 'unacc' 'acc' 'unacc' 'acc'
 'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc'
 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc' 'acc' 'acc'
 'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'unacc'
 'unacc' 'acc' 'unacc' 'unacc' 'acc' 'acc' 'unacc' 'unacc' 'unacc' 'unacc'
 'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'acc'
 'unacc' 'acc' 'acc' 'acc' 'unacc' 'acc' 'acc' 'unacc' 'acc' 'unacc' 'acc'
 'unacc' 'acc' 'unacc' 'unacc' 'acc' 'acc' 'unacc' 'unacc' 'unacc' 'unacc'
 'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'acc' 'acc'
 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc' 'unacc'
 'unacc' 'unacc' 'acc' 'unacc' 'acc' 'acc' 'acc' 'unacc' 'unacc' 'acc'
 'unacc' 'acc' 'acc' 'acc' 'unacc' 'unacc' 'acc' 'acc' 'acc' 'unacc' 'acc'
 'acc' 'unacc' 'unacc' 'acc' 'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc'
 'acc' 'unacc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'unacc'
 'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'acc' 'unacc' 'unacc'
 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc'
```

```
'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc'
'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'unacc' 'acc' 'unacc' 'unacc'
'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc'
'unacc' 'unacc' 'unacc' 'acc' 'unacc' 'acc' 'acc' 'acc' 'unacc' 'unacc'
'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'acc' 'unacc' 'unacc' 'unacc'
'unacc' 'acc']
```

In [ ]:

```
1
```