1 - Imports

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

2 - Loading the dat

```
from sklearn.datasets import fetch_openml
car_data=fetch_openml(name='car',version=2,parser='auto')
type(car_data)
sklearn.utils._bunch.Bunch
car_data.details
```

```
{'id': '991',
 'name': 'car',
 'version': '2',
 'description_version': '1',
 'format': 'ARFF',
 'upload_date': '2014-10-04T22:44:31',
 'licence': 'Public',
 'url': 'https://api.openml.org/data/v1/download/53525/car.arff',
 'parquet_url': 'https://openml1.win.tue.nl/datasets/0000/0991/dataset_991.pq',
 'file_id': '53525',
 'default_target_attribute': 'binaryClass',
 'tag': ['Chemistry',
  'derived',
  'Life Science',
  'mythbusting_1',
  'study_1',
  'study_15',
  'study_20',
  'study_41',
  'study_7'],
 'visibility': 'public',
 'minio_url': 'https://openml1.win.tue.nl/datasets/0000/0991/dataset_991.pq',
 'status': 'active',
 'processing_date': '2020-11-20 20:17:54',
 'md5_checksum': '49c57b793eef1b8e55f297e5e019fdbf'}
```

```
car_data.details['version']
```

```
'2'
```

```
print(car_data.DESCR)
```

```
**Author**:
**Source**: Unknown - Date unknown
**Please cite**:

Binarized version of the original data set (see version 1). The multi-class target feature is converted to a two-class nominal target feature by re-labeling the majority class as posi
```

```
Downloaded from openml.org.
```

```
car_data.feature_names
```

```
['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety']
```

```
car_data = car_data.frame
car_data.head()
```

	buying	maint	doors	persons	lug_boot	safety	binaryClass
0	vhigh	vhigh	2	2	small	low	P
1	vhigh	vhigh	2	2	small	med	P
2	vhigh	vhigh	2	2	small	high	P
3	vhigh	vhigh	2	2	med	low	P
4	vhigh	vhigh	2	2	med	med	P

Next steps: Generate code with car_data | View recommended plots | New interactive sheet

```
type(car_data)
```

```
pandas.core.frame.DataFrame
def __init__(data=None, index: Axes | None=None, columns: Axes | None=None, dtype: Dtype |
None=None, copy: bool | None=None) -> None

/usr/local/lib/python3.10/dist-packages/pandas/core/frame.py
Two-dimensional, size-mutable, potentially heterogeneous tabular data.

Data structure also contains labeled axes (rows and columns).
Arithmetic operations align on both row and column labels. Can be
thought of as a dict-like container for Series objects. The primary
```

3 - Exploratory Analysis

Before doing exploratory analysis, let's get the training and test data.

```
from sklearn.model_selection import train_test_split
train_data, test_data = train_test_split(car_data, test_size=0.3,
                                         random_state=20)
print('The size of training data is: {}\n The size of testing data is: {}'.format(len(train_data),
                                        len(test_data)))
```

```
The size of training data is: 1209
 The size of testing data is: 519
```

Checking Summary Statistics

```
train_data.describe()
```

```
        buying  maint  doors  persons  lug_boot  safety  binaryClass
count     1209   1209   1209     1209      1209    1209         1209
unique       4      4      4        3         3       3            2
top        med   high  5more     more       big     med            P
freq       327    311    319      418       411     406          849
```

Checking Missing Values

```
train_data.isnull().sum()
```

```
             0
buying       0
maint        0
doors        0
persons      0
lug_boot     0
safety       0
binaryClass  0

dtype: int64
```

Checking Categorical Features

Let's inspect some categorical features that are in the dataset, almost all . Let's see that!

```
train_data['buying'].value_counts()
```

```
        count
buying
med       327
high      307
vhigh     291
low       284

dtype: int64
```

```
train_data['maint'].value_counts()
```

```
       count
maint
 high    311
 med     311
vhigh    294
 low     293

dtype: int64
```

```
train_data['doors'].value_counts()
```

```
       count
doors
5more    319
  2      312
  4      296
  3      282

dtype: int64
```

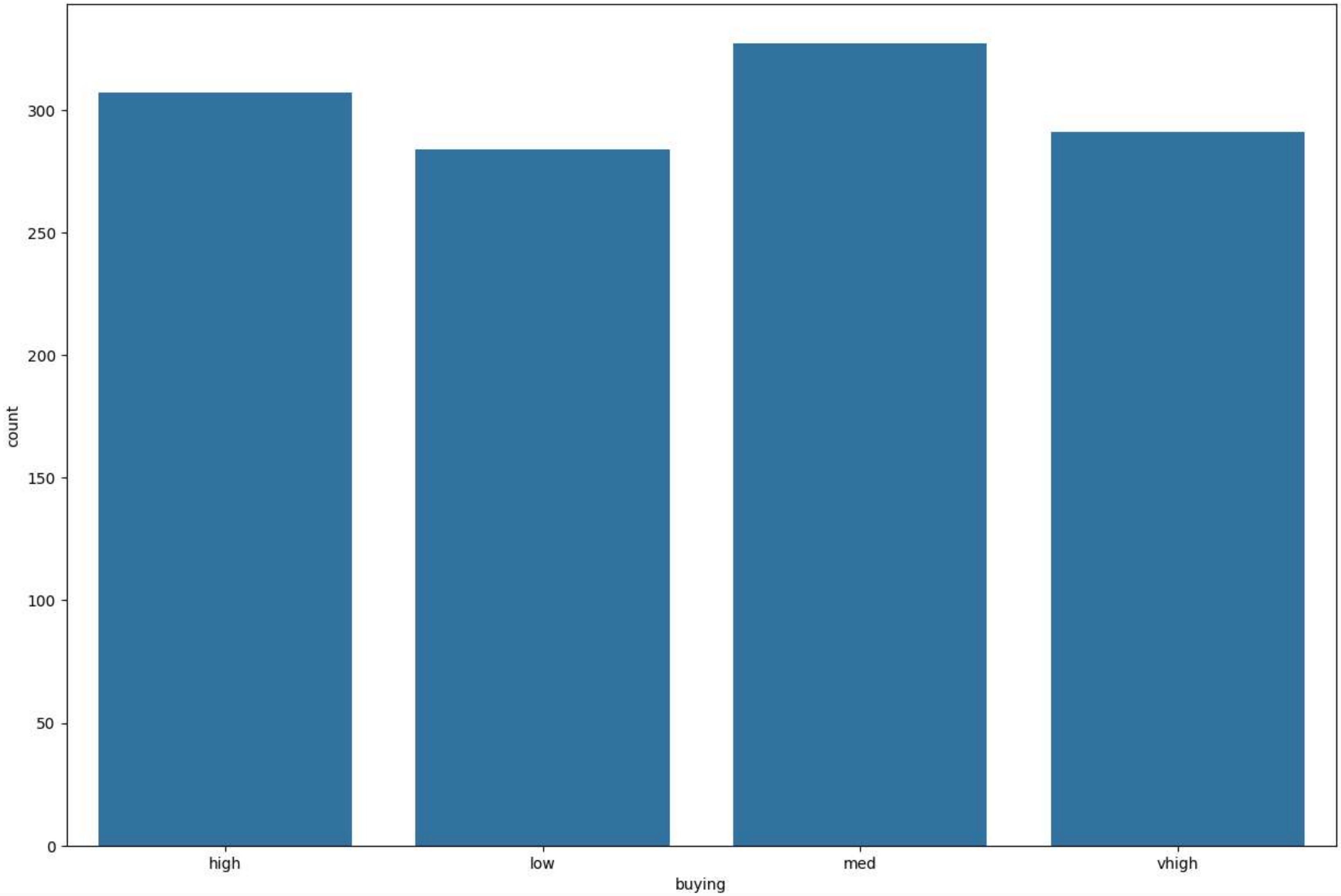
```
train_data['persons'].value_counts()
```

```
         count
persons
  more     418
   2       407
   4       384

dtype: int64
```

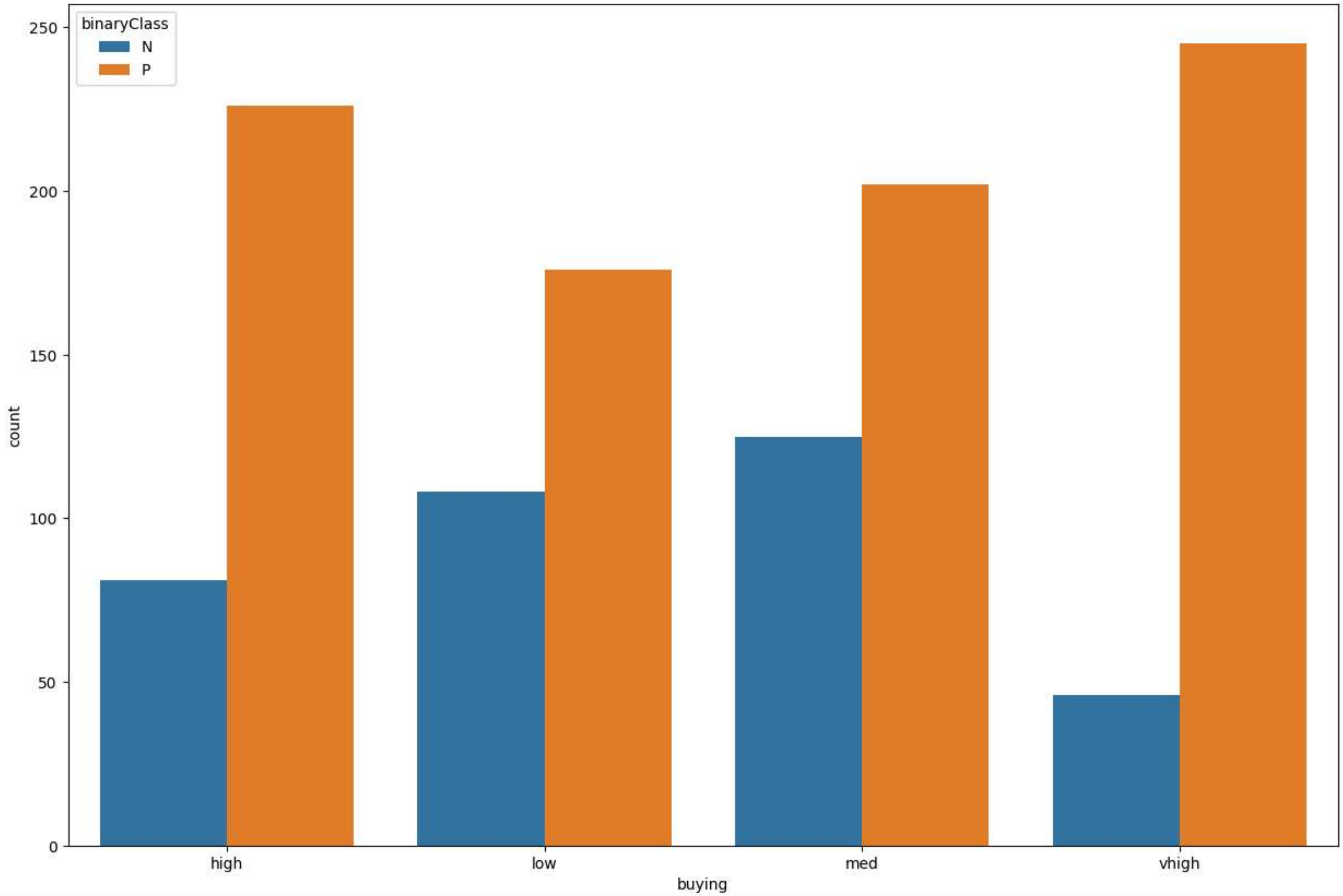
```
plt.figure(figsize=(15,10))
sns.countplot(data=train_data, x='buying')
```

```
<Axes: xlabel='buying', ylabel='count'>
```



```
plt.figure(figsize=(15,10))
sns.countplot(data=train_data, x='buying', hue='binaryClass')
```

```
<Axes: xlabel='buying', ylabel='count'>
```



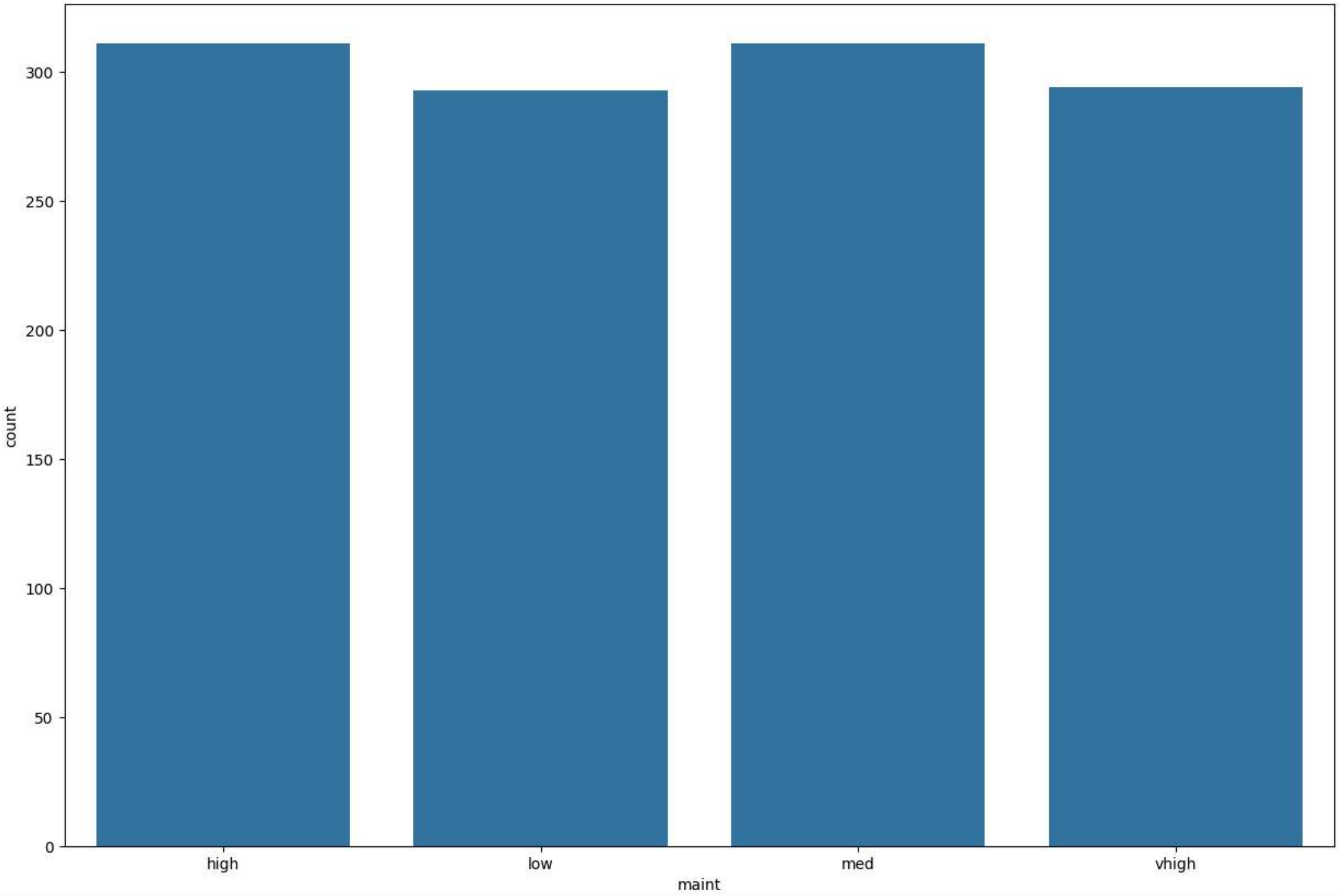
```
train_data['maint'].value_counts()
```

```
       count
maint
high     311
med      311
vhigh    294
low      293

dtype: int64
```

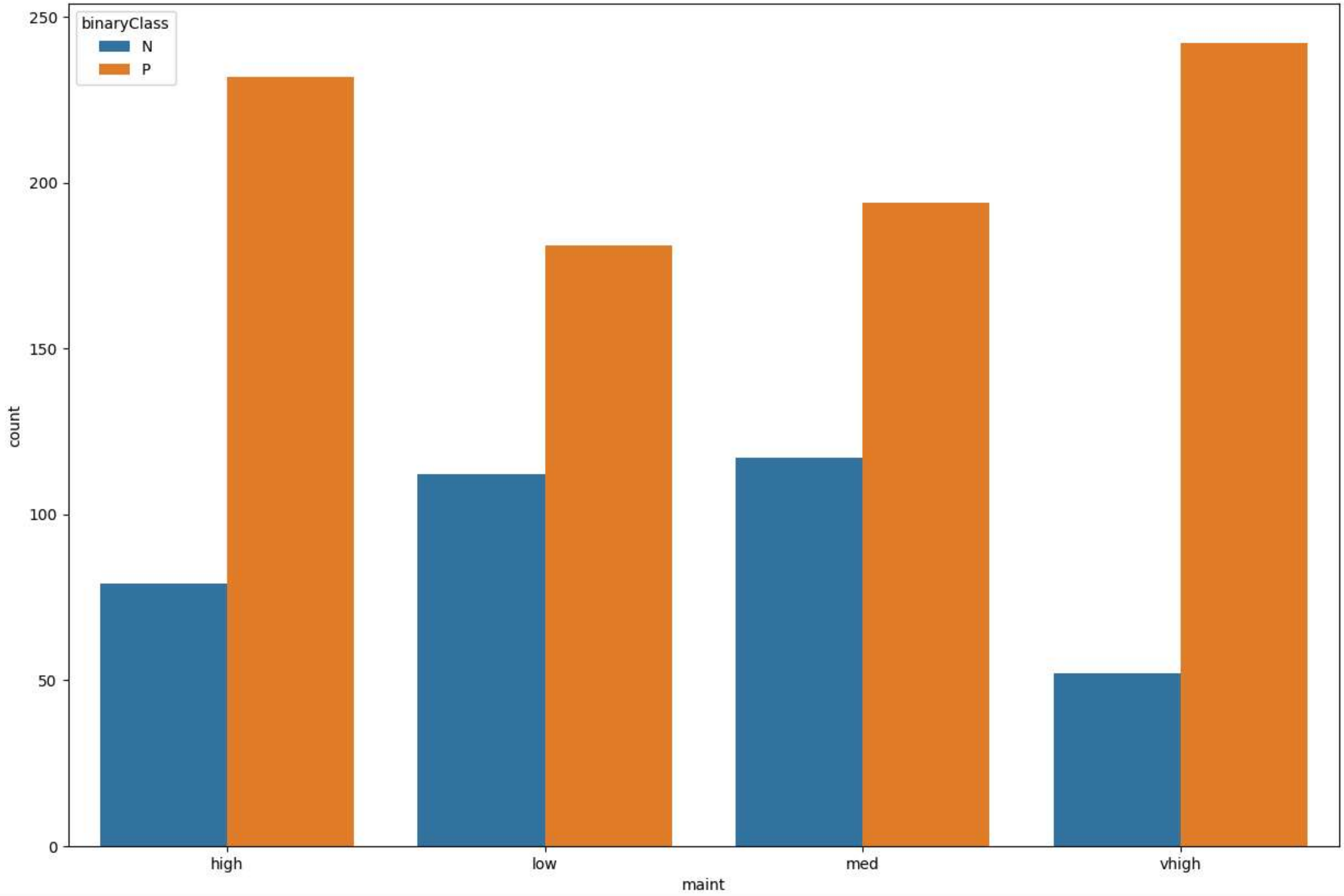
```
plt.figure(figsize=(15,10))
sns.countplot(data=train_data, x='maint')
```

```
<Axes: xlabel='maint', ylabel='count'>
```



```
plt.figure(figsize=(15,10))
sns.countplot(data=train_data, x='maint', hue='binaryClass')
```

```
<Axes: xlabel='maint', ylabel='count'>
```



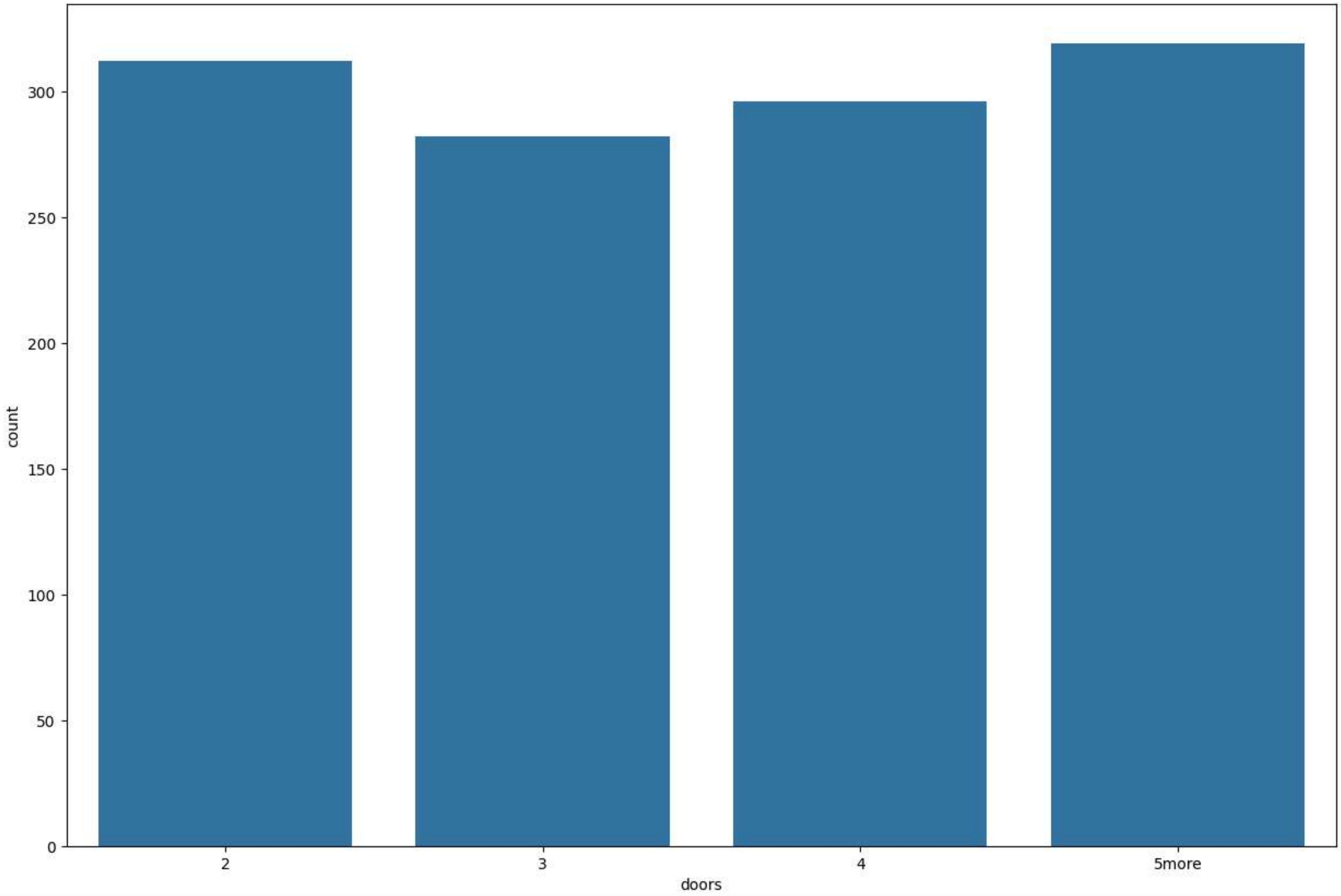
```
train_data['doors'].value_counts()
```

```
       count
doors
5more    319
    2    312
    4    296
    3    282

dtype: int64
```

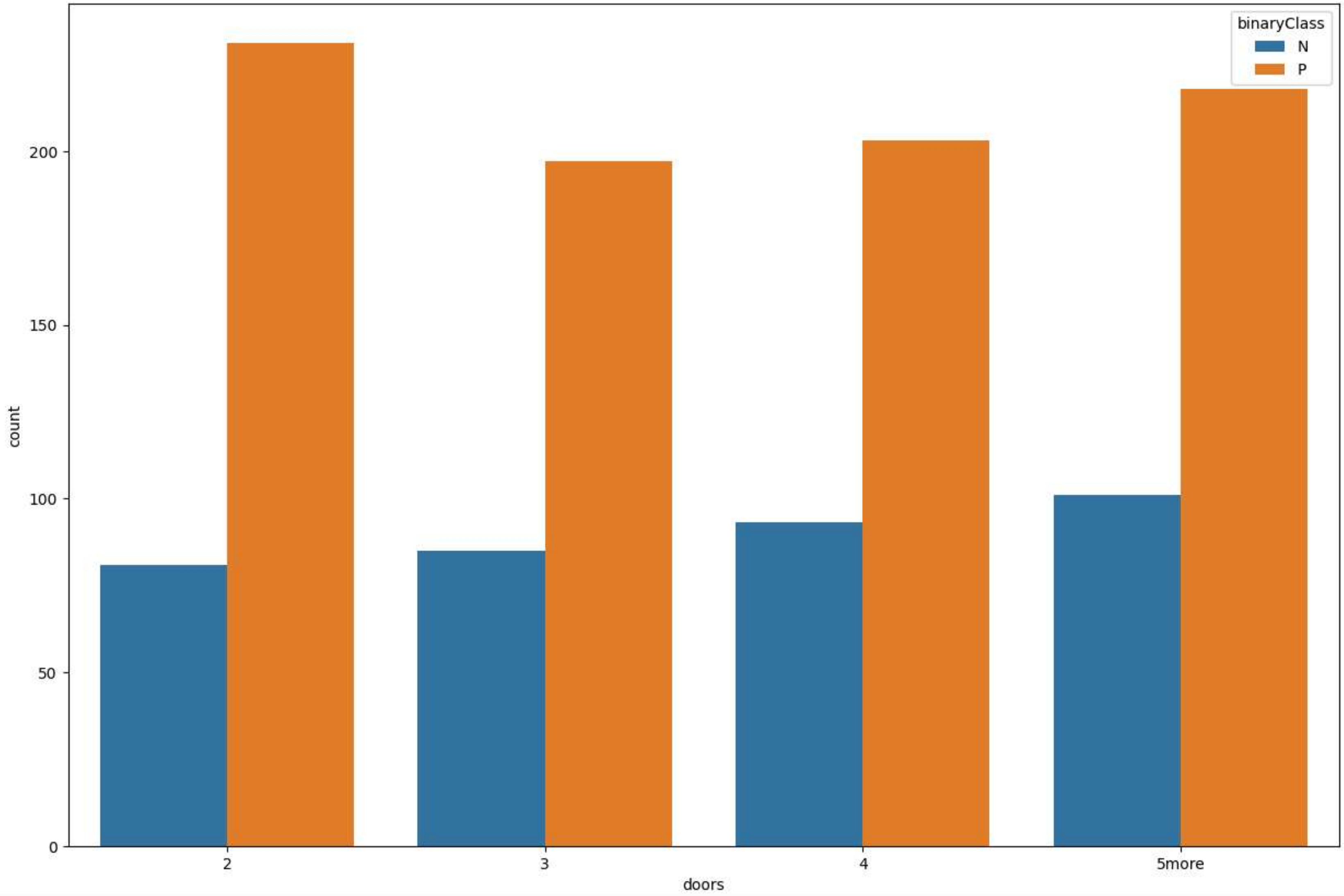
```
plt.figure(figsize=(15,10))
sns.countplot(data=train_data, x='doors')
```

```
<Axes: xlabel='doors', ylabel='count'>
```



```
plt.figure(figsize=(15,10))
sns.countplot(data=train_data, x='doors', hue='binaryClass')
```

```
<Axes: xlabel='doors', ylabel='count'>
```



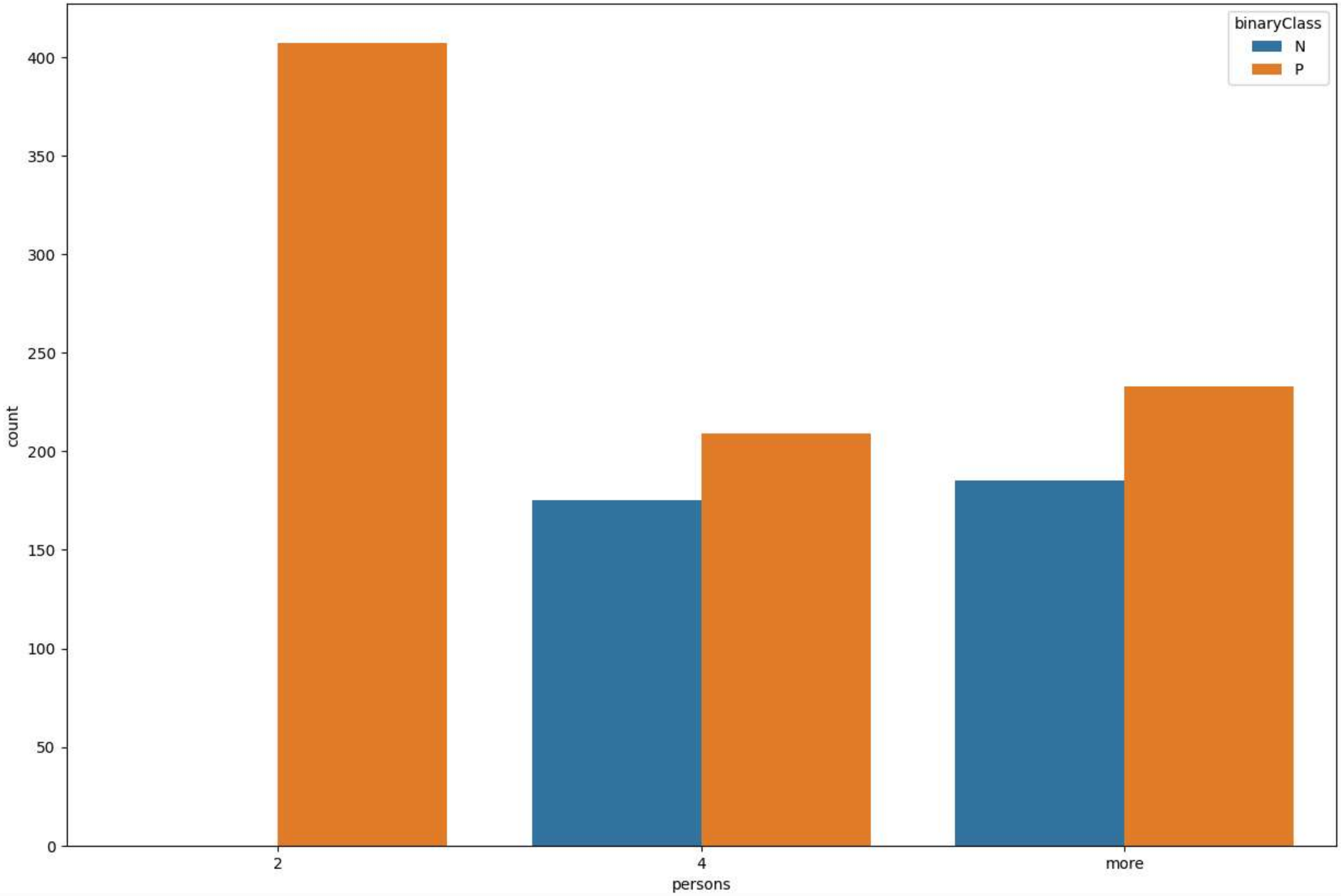
```
train_data['persons'].value_counts()
```

```
          count
persons
   more     418
      2     407
      4     384

dtype: int64
```

```
plt.figure(figsize=(15,10))
sns.countplot(data=train_data, x='persons', hue='binaryClass')
```

```
<Axes: xlabel='persons', ylabel='count'>
```



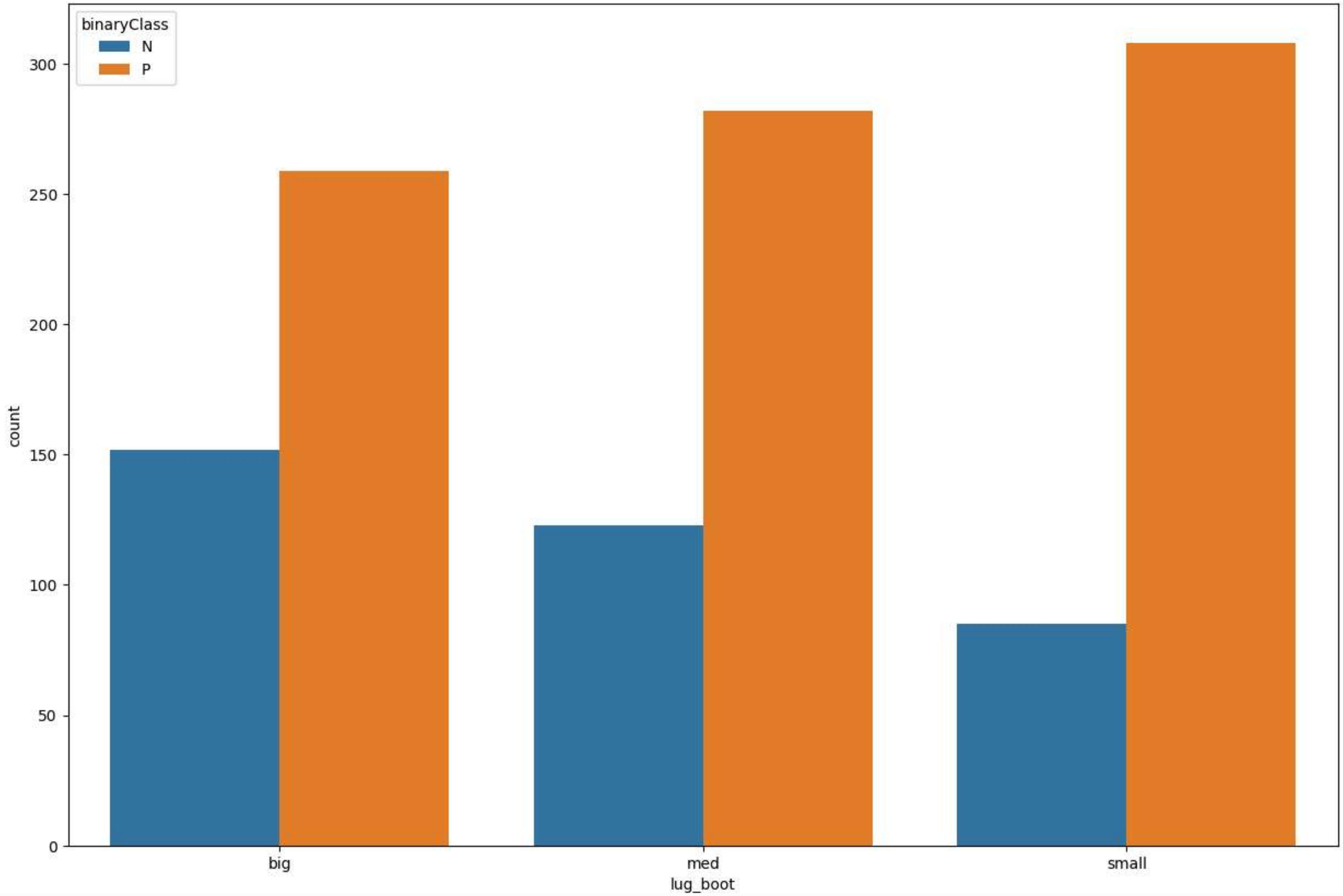
```
train_data['lug_boot'].value_counts()
```

```
          count
lug_boot
big         411
med         405
small       393

dtype: int64
```

```
plt.figure(figsize=(15,10))
sns.countplot(data=train_data, x='lug_boot', hue='binaryClass')
```

```
<Axes: xlabel='lug_boot', ylabel='count'>
```



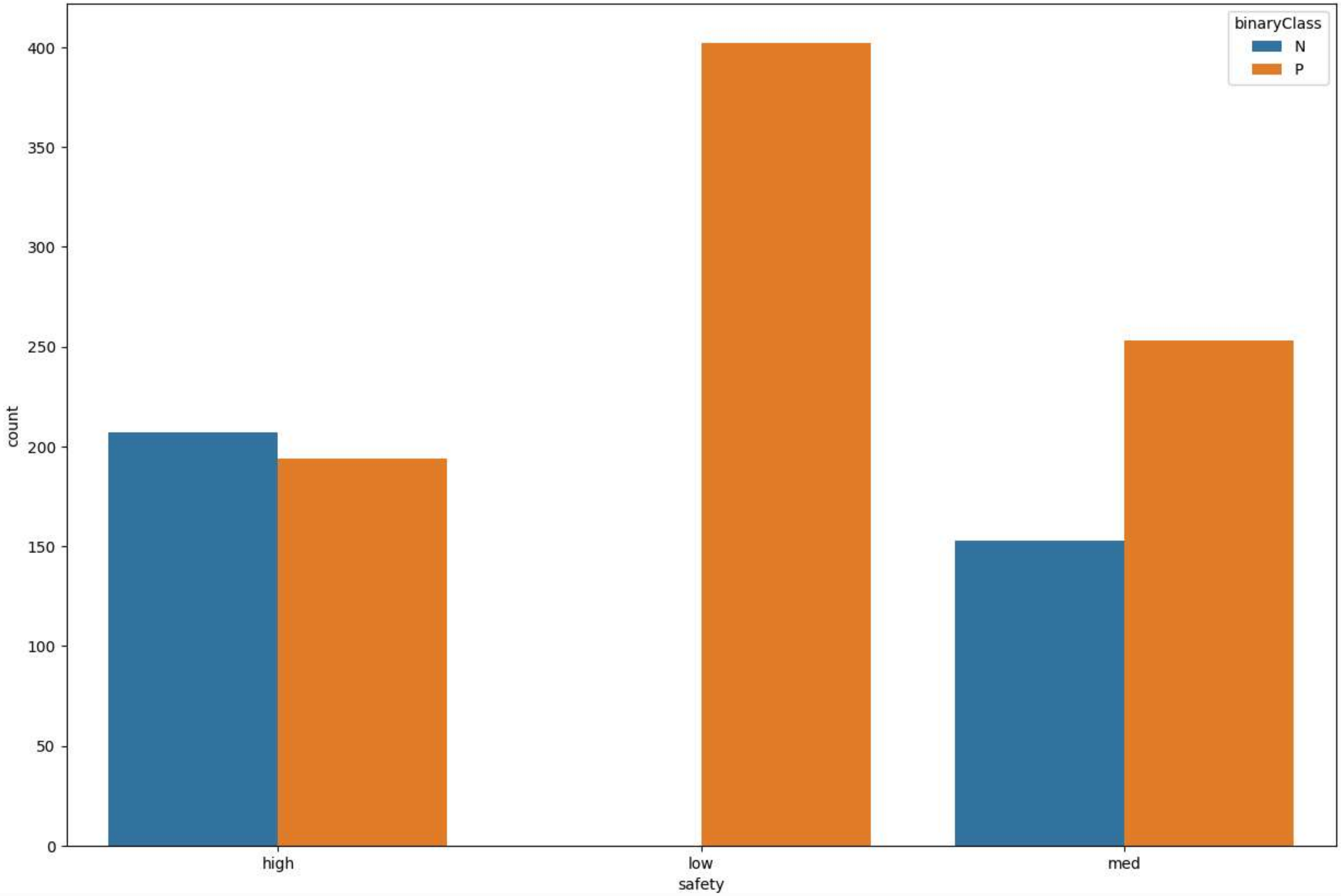
```
train_data['safety'].value_counts()
```

```
        count
safety
 med      406
 low      402
 high     401

dtype: int64
```

```
plt.figure(figsize=(15,10))
sns.countplot(data=train_data, x='safety', hue='binaryClass')
```

```
<Axes: xlabel='safety', ylabel='count'>
```



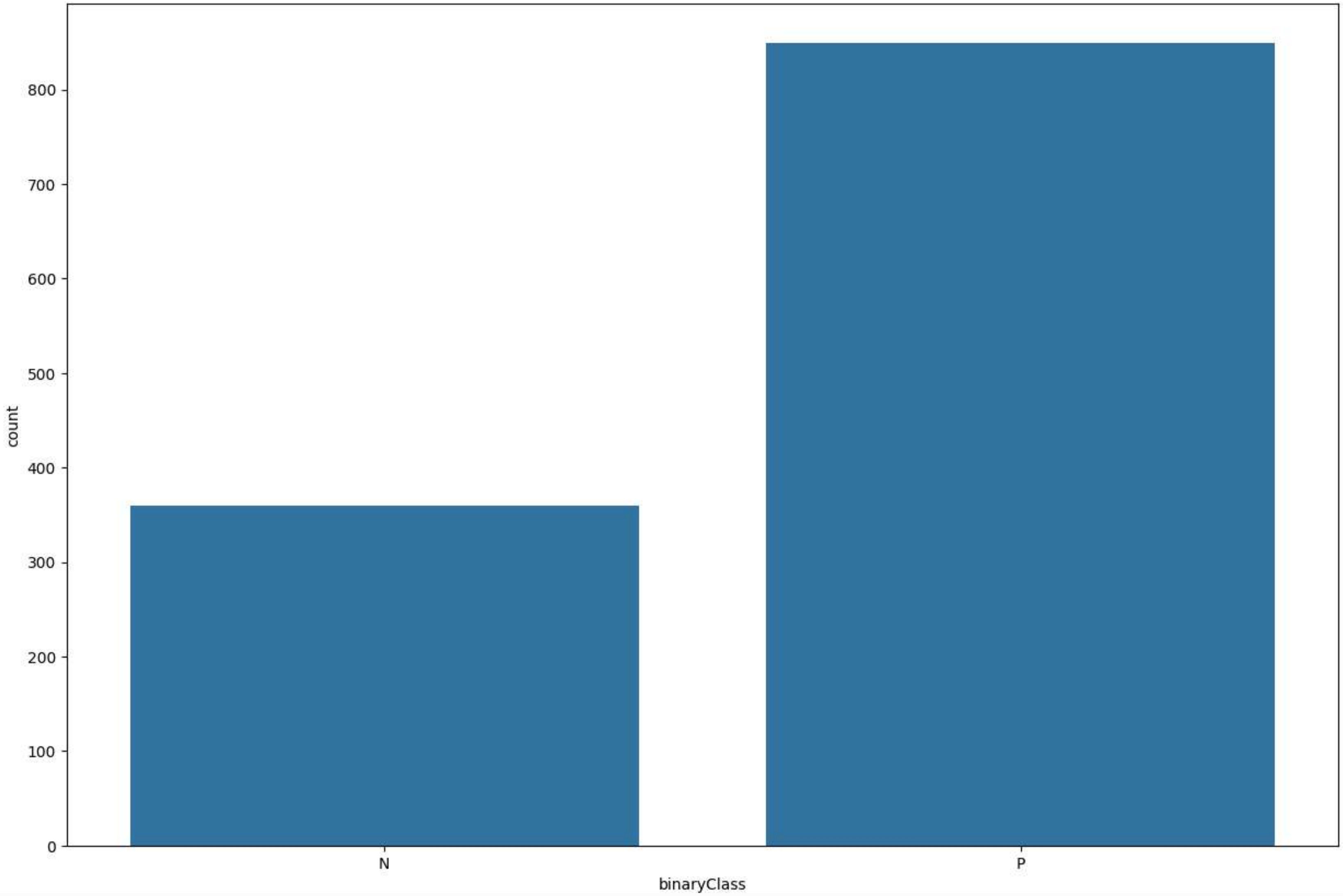
```
train_data['binaryClass'].value_counts()
```

```
                 count
binaryClass
     P            849
     N            360

dtype: int64
```

```
plt.figure(figsize=(15,10))
sns.countplot(data=train_data, x='binaryClass')
```

```
<Axes: xlabel='binaryClass', ylabel='count'>
```



4 - Data Preprocessing

Handling Categorical Features

```
car_train =train_data.drop('binaryClass', axis=1)
car_labels = train_data[['binaryClass']]
```

```
from sklearn.preprocessing import OrdinalEncoder
from sklearn.pipeline import Pipeline
pipe = Pipeline([('ord_enc', OrdinalEncoder())])
car_train_prepared = pipe.fit_transform(car_train)
from sklearn.preprocessing import LabelEncoder
label_enc = LabelEncoder()
car_labels_prepared = label_enc.fit_transform(car_labels)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape
  y = column_or_1d(y, warn=True)
```

5 - Training Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
tree_clf = DecisionTreeClassifier()
tree_clf.fit(car_train_prepared, car_labels_prepared)
```

```
▾ DecisionTreeClassifier ⓘ ?
DecisionTreeClassifier()
```

```
from sklearn.tree import DecisionTreeClassifier, export_text # Import the export_text function
tree_clf = DecisionTreeClassifier()
tree_clf.fit(car_train_prepared, car_labels_prepared)

text_representation = export_text(tree_clf) # Call export_text with tree_clf
print(text_representation)
```

```
|   |   |   |   |   |--- feature_0 >  2.50
|   |   |   |   |   |   |--- feature_1 <= 0.50
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- feature_1 >  0.50
|   |   |   |   |   |   |   |--- feature_4 <= 0.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- feature_4 >  0.50
|   |   |   |   |   |   |   |   |--- feature_2 <= 1.50
|   |   |   |   |   |   |   |   |   |--- feature_2 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- feature_2 >  0.50
```