In [1]:

```
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
data=pd.read_csv("d:/datasets/car.data",names=["buying","maint","doors","persons","lug_boot","safety
data.head()
```

Out[1]:

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

In [2]:

```
data.dropna(inplace=True)
data=pd.get_dummies(data,columns=["doors","persons","buying","maint","lug_boot","safety"])
data["car_Eva"]=data["car_Eva"].map({"unacc":0,"acc":1,"good":2,"vgood":3})
X=data.drop(["car_Eva"],axis=1)
y=data["car_Eva"]
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=10)
```

In [3]:

```
from sklearn.linear_model import LogisticRegression
clf=LogisticRegression(random_state=10,class_weight="balanced")
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
```

In [4]:

```
clf.score(X_test,y_test)
```

Out[4]:

```
0.9036608863198459
```

In [5]:

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

```
              precision    recall  f1-score   support

           0       1.00      0.91      0.95       371
           1       0.74      0.84      0.79       102
           2       0.58      0.90      0.70        21
           3       0.81      1.00      0.89        25

    accuracy                           0.90       519
   macro avg       0.78      0.92      0.84       519
weighted avg       0.92      0.90      0.91       519
```

In [6]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred,normalize=False)
```

Out[6]:

```
469
```

In [7]:

```
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

In [8]:

```
precision_score(y_test,y_pred,average="macro")
```

Out[8]:

```
0.7808971247514074
```

In [9]:

```
recall_score(y_test,y_pred,average="macro")
```

Out[9]:

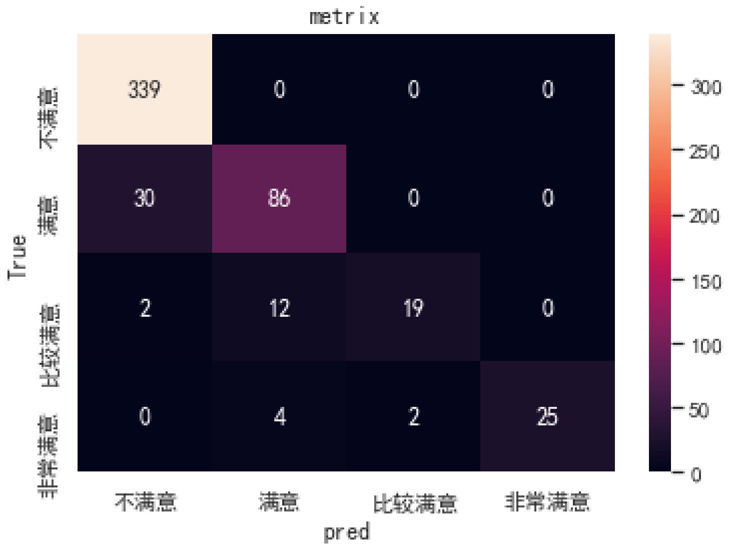
```
0.915411447597907
```

In [10]:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
sns.set(font="SimHei")
ax=sns.heatmap(confusion_matrix(y_pred,y_test),annot=True,fmt="d",
               xticklabels=["不满意","满意","比较满意","非常满意"],yticklabels=["不满意","满意","比
ax.set_ylabel("True")
ax.set_xlabel("pred")
ax.set_title("metrix")
```

Out[10]:

```
Text(0.5, 1.0, 'metrix')
```



In [11]:

```
print(classification_report(y_test,y_pred,labels=[0,1,2,3],target_names=["不满意","满意","比较满意",
```

```
              precision    recall  f1-score   support

         不满意       1.00      0.91      0.95       371
          满意       0.74      0.84      0.79       102
        比较满意       0.58      0.90      0.70        21
        非常满意       0.81      1.00      0.89        25

    accuracy                           0.90       519
   macro avg       0.78      0.92      0.84       519
weighted avg       0.92      0.90      0.91       519
```

In [12]:

```
from sklearn.metrics import auc
from sklearn.metrics import precision_recall_curve
```

In [13]:

```
import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_curve
```

In [14]:

```
clf.predict_proba(X_test)
```

Out[14]:

```
array([[9.97998681e-01, 1.99795877e-03, 3.35767663e-06, 2.28522035e-09],
       [9.99432677e-01, 5.67305027e-04, 1.82684668e-08, 3.16304855e-11],
       [9.97677550e-01, 2.29372995e-03, 2.85537414e-05, 1.66740429e-07],
       ...,
       [8.90657182e-04, 4.09949546e-02, 3.83018560e-03, 9.54284203e-01],
       [9.96992049e-01, 2.98035071e-03, 2.75823272e-05, 1.77907231e-08],
       [8.48406037e-02, 9.07884125e-01, 7.11246629e-03, 1.62804862e-04]])
```

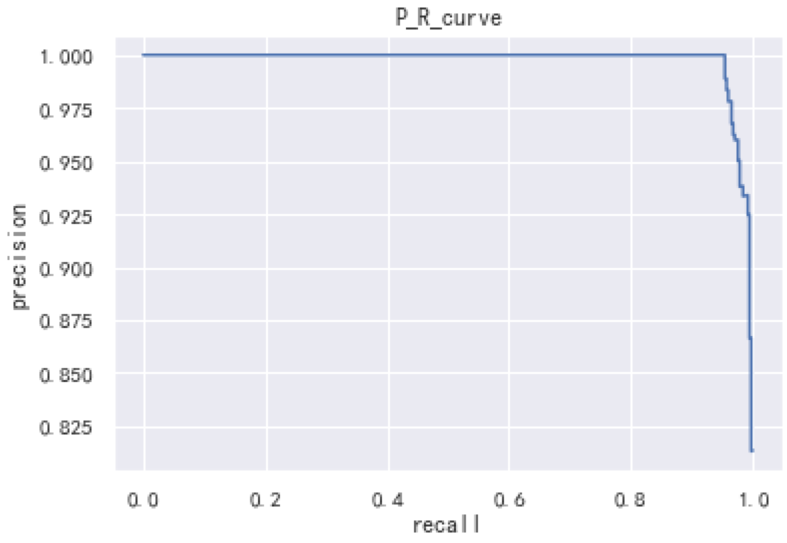
In [15]:

```
k=0  #k=0,"不满意",1,"满意",2,"比较满意",3,"非常满意"
probs=clf.predict_proba(X_test)[:,k]
precision,recall,thresholds=precision_recall_curve(y_test,probs,pos_label=k)

plt.plot(recall,precision)
plt.title("P_R_curve")
plt.xlabel("recall")
plt.ylabel("precision")
```

Out[15]:

```
Text(0, 0.5, 'precision')
```



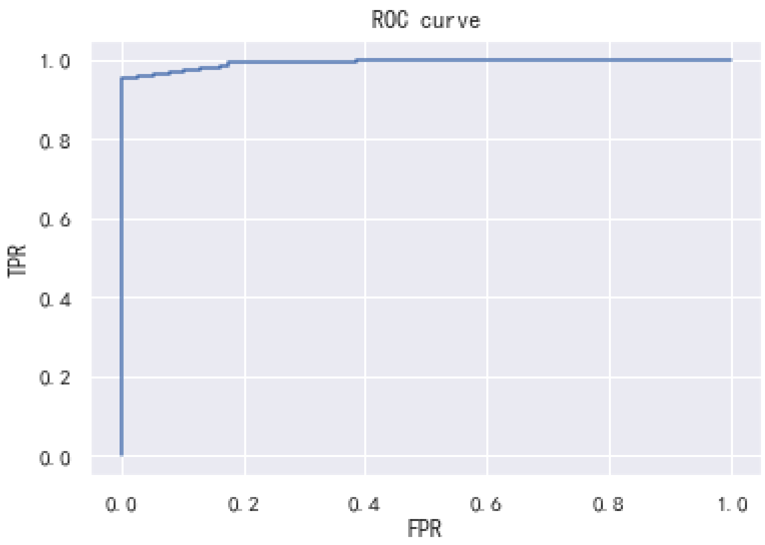
In [16]:

```
k=0  #k=0,"不满意",1,"满意",2,"比较满意",3,"非常满意"

from sklearn.metrics import roc_curve
probs=clf.predict_proba(X_test)[:,k]
fpr,tpr,thresholds=roc_curve(y_test,y_score=probs,pos_label=k)
plt.plot(fpr,tpr)
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC curve")
```

Out[16]:

```
Text(0.5, 1.0, 'ROC curve')
```



In [17]:

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import auc
```

In [18]:

```
#多分类计算AUC
probs=clf.predict_proba(X_test)
```

In [19]:

```
k=1  #k=0,"不满意",1,"满意",2,"比较满意",3,"非常满意"
roc_auc_score(y_test,probs,multi_class="ovr",average='macro')  #计算的平均
```

Out[19]:

```
0.9849370144081224
```

In [20]:

```
auc(fpr,tpr)   #计算K类，取决fpr,tpr的计算
```

Out[20]:

```
0.9927332993370729
```

In [23]:

```
#log_loss()  在二分类中
```

In [22]:

```
from sklearn.svm import SVC
from sklearn.metrics import hinge_loss
ksvm=SVC(random_state=10)
ksvm.fit(X_train,y_train)
pred_SVC=ksvm.decision_function(X_test)
hinge_loss(y_test,pred_SVC)
```

Out[22]:

```
0.053448825746343416
```

In [ ]:

```

```