广告投放和销售额数据分析

1.相关背景

（1）数据集简介

Advertising数据集包含有关产品在200个不同市场中的销售情况的统计信息，以及在每个市场中针对不同媒体渠道（电视，广播和报纸）的广告预算。销售单位是数千个，预算单位是数千美元。

（2）分析数据

自变量：广告支出，TV、radio、newspaper

因变量：商品销售额，sales

求解：上述三个因素对于商品销售额的回归模型

2.数据读取

调用Pandas的read_csv()函数，读取数据。

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os
data=pd.read_csv("Advertising.csv",header=0)
data.head()
```

Out[1]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

3.数据理解

（1）查看数据形状

In [2]:
```
data.shape
```

Out[2]: (200, 4)

（2）查看列名

In [3]:
```
data.columns
```

Out[3]: Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')

（3）查看数据信息

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

（4）查看描述性统计信息

In [5]:
```
data.describe()
```

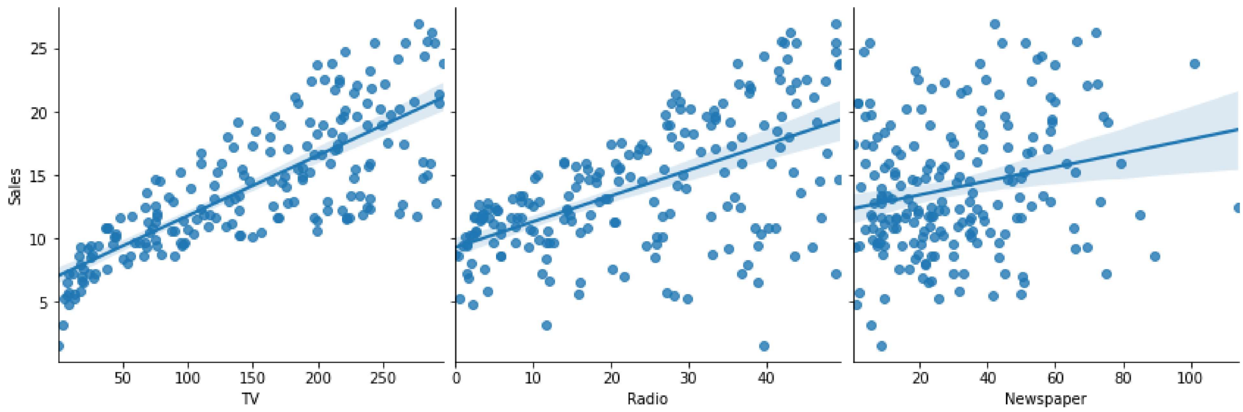
Out[5]:

	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	14.022500
std	85.854236	14.846809	21.778621	5.217457
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	10.375000
50%	149.750000	22.900000	25.750000	12.900000
75%	218.825000	36.525000	45.100000	17.400000
max	296.400000	49.600000	114.000000	27.000000

（5）数据可视化

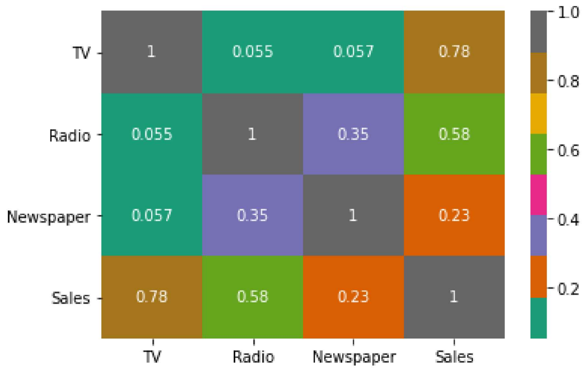
In [6]:
```
#带线性回归最佳拟合线的散点图

def target_scatter(dataframe):
    cols = [col for col in dataframe.columns if col != "Sales"]
    sns.pairplot(dataframe, x_vars=cols, y_vars="Sales", height=4, aspect=1, kind='reg')
    plt.show()
target_scatter(data)
```



In [7]:
```
# 相关图，直观地查看给定数据框（或二维数组）中所有可能的数值变量对之间的相关度量

sns.heatmap(data.corr(), cmap="Dark2", annot = True)
plt.show()
```



从可视化结果中可以看出，三个自变量TV，radio、newspaper均对因变量sales有显著的关系。

4.数据准备

构建特征矩阵Data和目标向量sales。

In [8]:
```python
Data=data.drop(['Sales'],axis=1)
Data.head()
```

Out[8]:

	TV	Radio	Newspaper
0	230.1	37.8	69.2
1	44.5	39.3	45.1
2	17.2	45.9	69.3
3	151.5	41.3	58.5
4	180.8	10.8	58.4

In [9]:
```python
#修改目标向量为后续statsmodels包中所要求的类型

Sales=data['Sales']
import numpy as np
Sales=np.ravel(Sales)
type(Sales)
```

Out[9]:
```
numpy.ndarray
```

5.构建模型

In [10]:
```python
import statsmodels.api as sm
```

给Data新增一列，列名X_add_const，每行取值1.0。

In [11]:
```python
X_add_const=sm.add_constant(Data.to_numpy())
X_add_const
```

Out[11]:
```
array([[  1. , 230.1,  37.8,  69.2],
       [  1. ,  44.5,  39.3,  45.1],
       [  1. ,  17.2,  45.9,  69.3],
       [  1. , 151.5,  41.3,  58.5],
       [  1. , 180.8,  10.8,  58.4],
       [  1. ,   8.7,  48.9,  75. ],
       [  1. ,  57.5,  32.8,  23.5],
       [  1. , 120.2,  19.6,  11.6],
       [  1. ,   8.6,   2.1,   1. ],
       [  1. , 199.8,   2.6,  21.2],
       [  1. ,  66.1,   5.8,  24.2],
       [  1. , 214.7,  24. ,   4. ],
       [  1. ,  23.8,  35.1,  65.9],
       [  1. ,  97.5,   7.6,   7.2],
       [  1. , 204.1,  32.9,  46. ],
       [  1. , 195.4,  47.7,  52.9],
       [  1. ,  67.8,  36.6, 114. ],
       [  1. , 281.4,  39.6,  55.8],
       [  1. ,  69.2,  20.5,  18.3],
       [  1. , 147.3,  23.9,  19.1],
```

In [12]:
```python
myModel=sm.OLS(Sales,X_add_const)
```

训练具体模型:

In [13]:
```
results=myModel.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.897
Model:                            OLS   Adj. R-squared:                  0.896
Method:                 Least Squares   F-statistic:                     570.3
Date:                Sun, 13 Mar 2022   Prob (F-statistic):           1.58e-96
Time:                        01:37:41   Log-Likelihood:                -386.18
No. Observations:                 200   AIC:                             780.4
Df Residuals:                     196   BIC:                             793.6
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.9389      0.312      9.422      0.000       2.324       3.554
x1             0.0458      0.001     32.809      0.000       0.043       0.049
x2             0.1885      0.009     21.893      0.000       0.172       0.206
x3            -0.0010      0.006     -0.177      0.860      -0.013       0.011
==============================================================================
Omnibus:                       60.414   Durbin-Watson:                   2.084
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              151.241
Skew:                          -1.327   Prob(JB):                     1.44e-33
Kurtosis:                       6.332   Cond. No.                         454.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

6.模型预测

通过调用statsmodel包中的predict()函数，基于自变量X对因变量y进行预测。

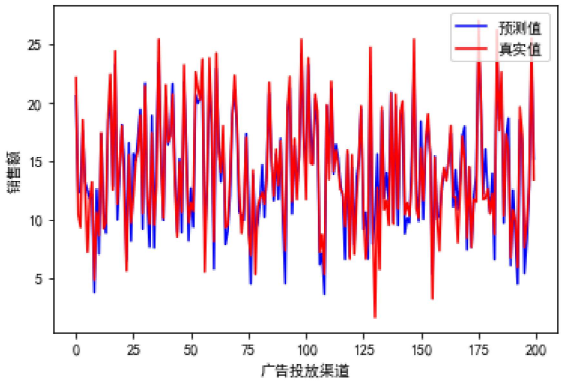
In [14]:
```
y_pred=results.predict(X_add_const)
y_pred[0:5]
```

Out[14]:
```
array([20.52397441, 12.33785482, 12.30767078, 17.59782951, 13.18867186])
```

7.模型评价

数据可视化方法显示真实值和拟合值之间的差异

In [15]:
```
import matplotlib.pyplot as plt
plt.figure()
plt.rcParams['font.family']='simHei'
plt.plot(range(len(y_pred)),y_pred,'blue',label="预测值")
plt.plot(range(len(y_pred)),Sales,'red',label="真实值")
plt.legend(loc="upper right")
plt.xlabel("广告投放渠道")
plt.ylabel("销售额")
plt.show()
```



医疗保险费用影响因素分析

1.相关背景

（1）数据集简介

Insurance数据集包含了1138条医疗保险数据，包含年龄、性别、BMI指数等相关数据，这些因素都有可能会影响到医疗保险的费用。以下是数据集中的变量解释：

- **age:** age of primary beneficiary
- **sex:** insurance contractor gender, female, male
- **bmi:** Body mass index, providing an understanding of body, weights that are relatively high or low relative to height, objective index of body weight (kg / m ^ 2) using the ratio of height to weight, ideally 18.5 to 24.9
- **children:** Number of children covered by health insurance / Number of dependents
- **smoker:** Smoking
- **region:** the beneficiary's residential area in the US, northeast, southeast, southwest, northwest

（2）分析数据

自变量：age, sex, bmi, children, smoker, region

因变量：charge

求解：哪种因素对保险费的影响最大？

2.数据读取

调用Pandas的read_csv()函数，读取数据。

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import os
data=pd.read_csv("insurance.csv",header=0)
data.head()
```

Out[1]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

3.数据理解

（1）查看数据形状

In [2]:
```
data.shape
```

Out[2]: (1338, 7)

（2）查看列名

In [3]:
```
data.columns
```

Out[3]: Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dtype='object')

（3）查看数据信息

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1338 non-null   int64  
 1   sex       1338 non-null   object 
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64  
 4   smoker    1338 non-null   object 
 5   region    1338 non-null   object 
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

（4）查看描述性统计信息

In [5]:
```
data.describe()
```

Out[5]:

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

（5）查看有无空值

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

（6）查看有无重复值

In [7]:
```
data.duplicated().sum()
```

Out[7]:
```
1
```

删除重复值:

In [8]:
```
data.drop_duplicates(inplace=True)
data.duplicated()
```

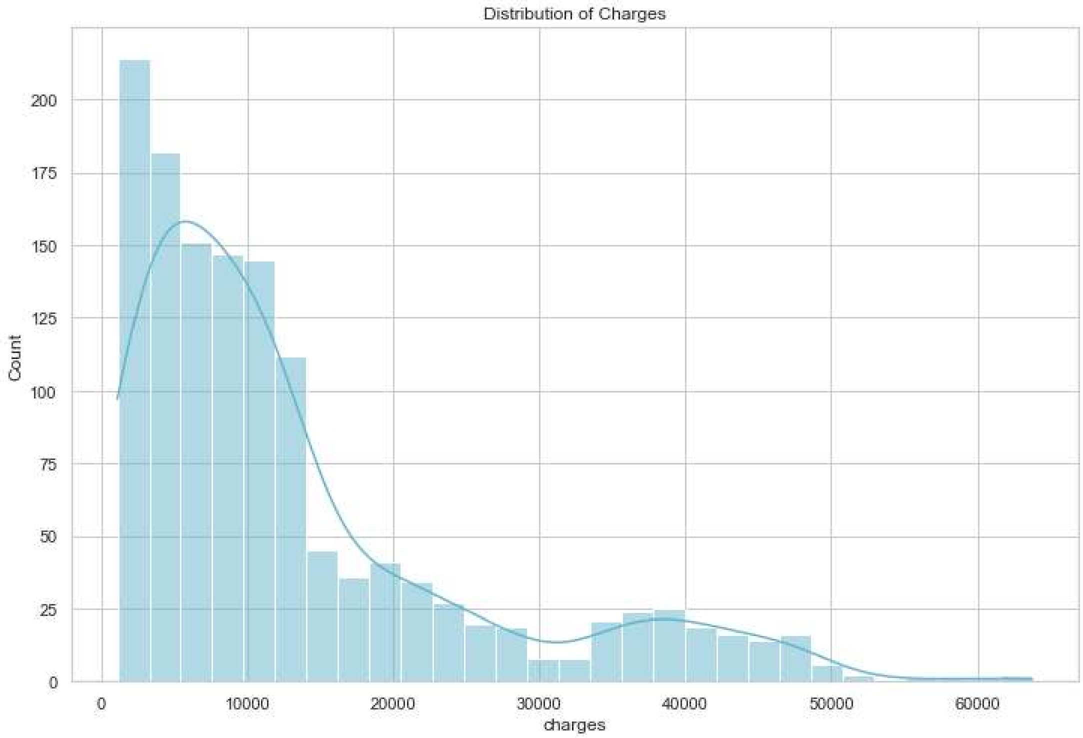
Out[8]:
```
0       False
1       False
2       False
3       False
4       False
        ...  
1333    False
1334    False
1335    False
1336    False
1337    False
Length: 1337, dtype: bool
```

（7）数据可视化

保险费用的分布图:

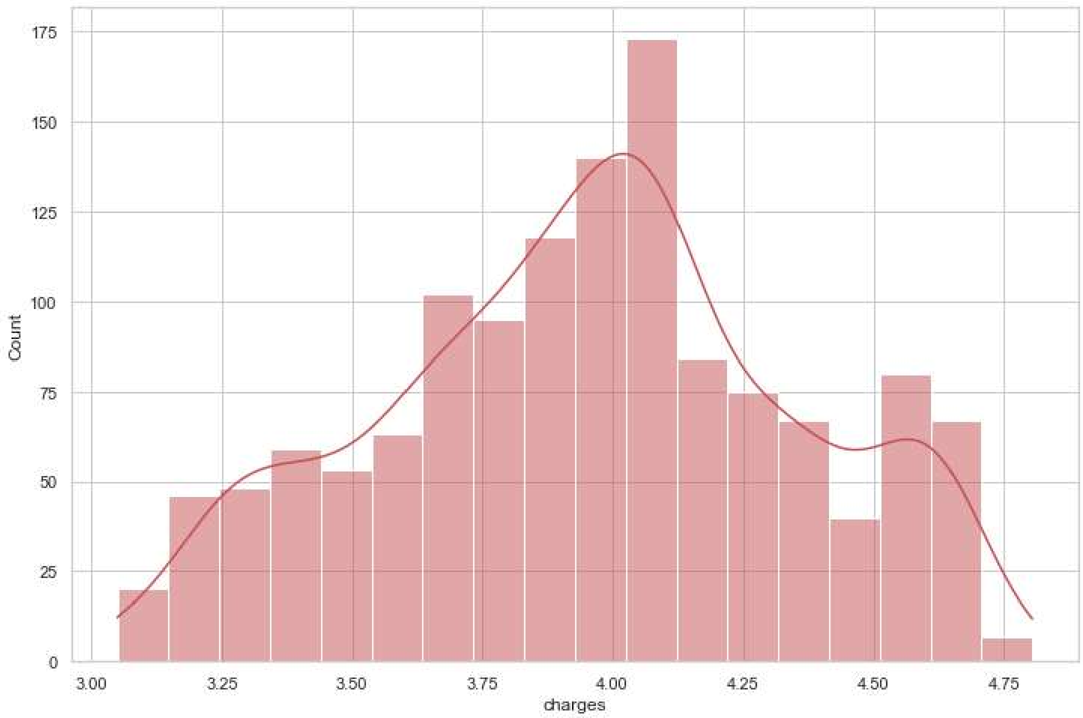
In [9]:
```
sns.set(style='whitegrid')
f,ax = plt.subplots(1,1,figsize=(12, 8))
ax = sns.histplot(data['charges'], kde = True, color = 'c')
plt.title('Distribution of Charges')
```

Out[9]: Text(0.5, 1.0, 'Distribution of Charges')



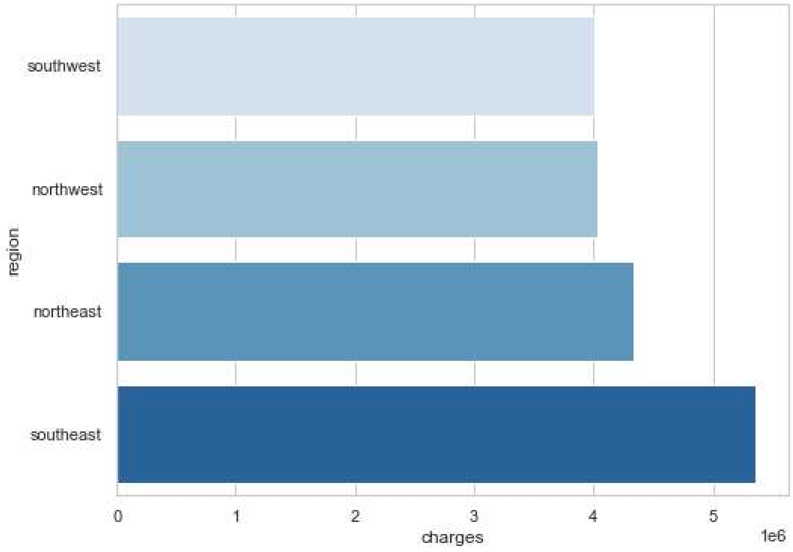
由于图像是右偏的，所以我们将它log化:

In [10]:
```
f, ax = plt.subplots(1, 1, figsize=(12, 8))
ax = sns.histplot(np.log10(data['charges']), kde = True, color = 'r' )
```



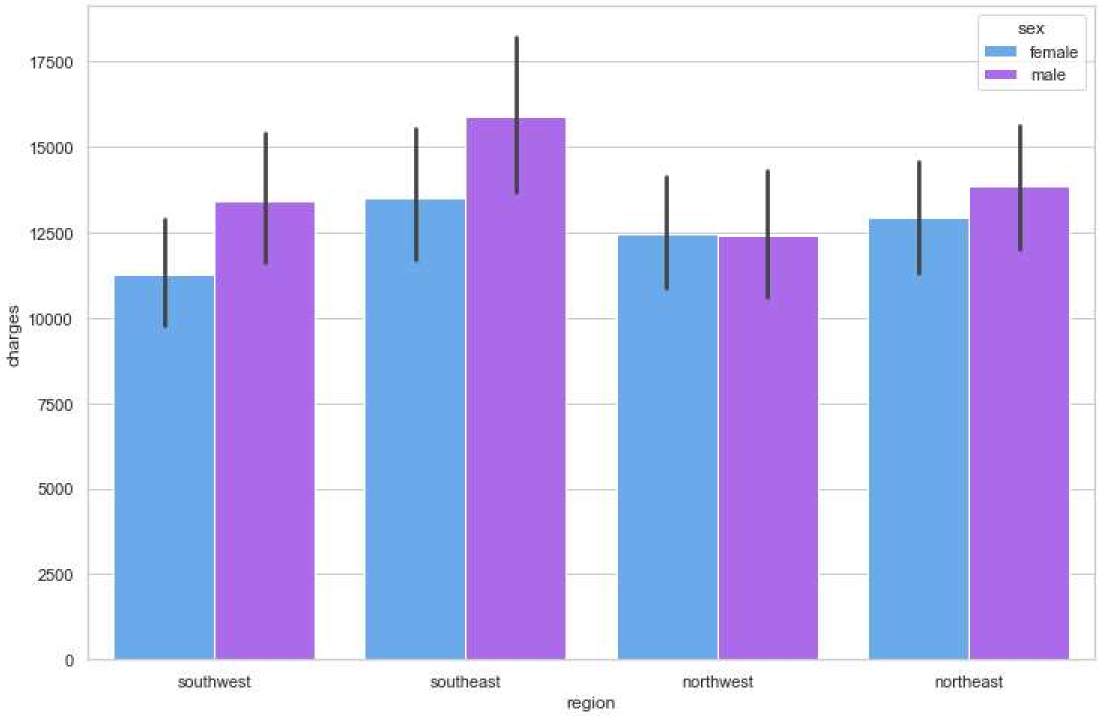
下一步我们来看不同区域region的费用分布图:

In [11]:
```
charges = data['charges'].groupby(data.region).sum().sort_values(ascending = True)
f, ax = plt.subplots(1, 1, figsize=(8, 6))
ax = sns.barplot(x=charges.head(), y=charges.head().index, palette='Blues')
```

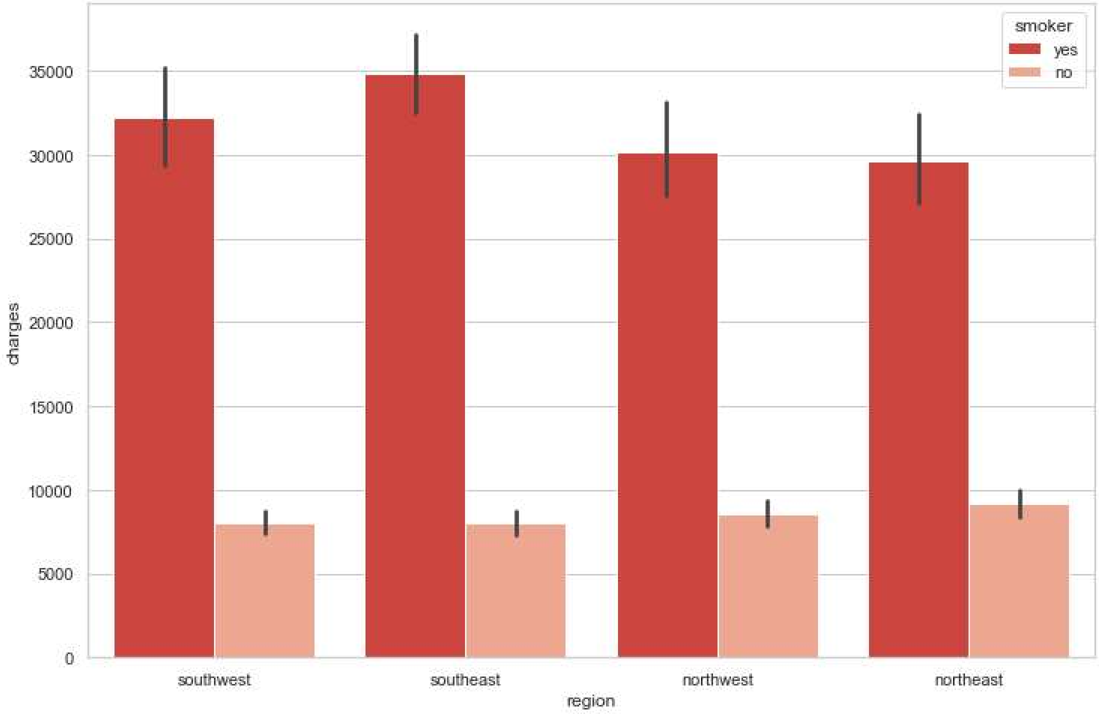


所以总的来说，医疗费用最高的是在东南部，最低的是在西南部。考虑到某些因素(性别、吸烟、有孩子)，让我们看看它在不同地区是如何变化的:

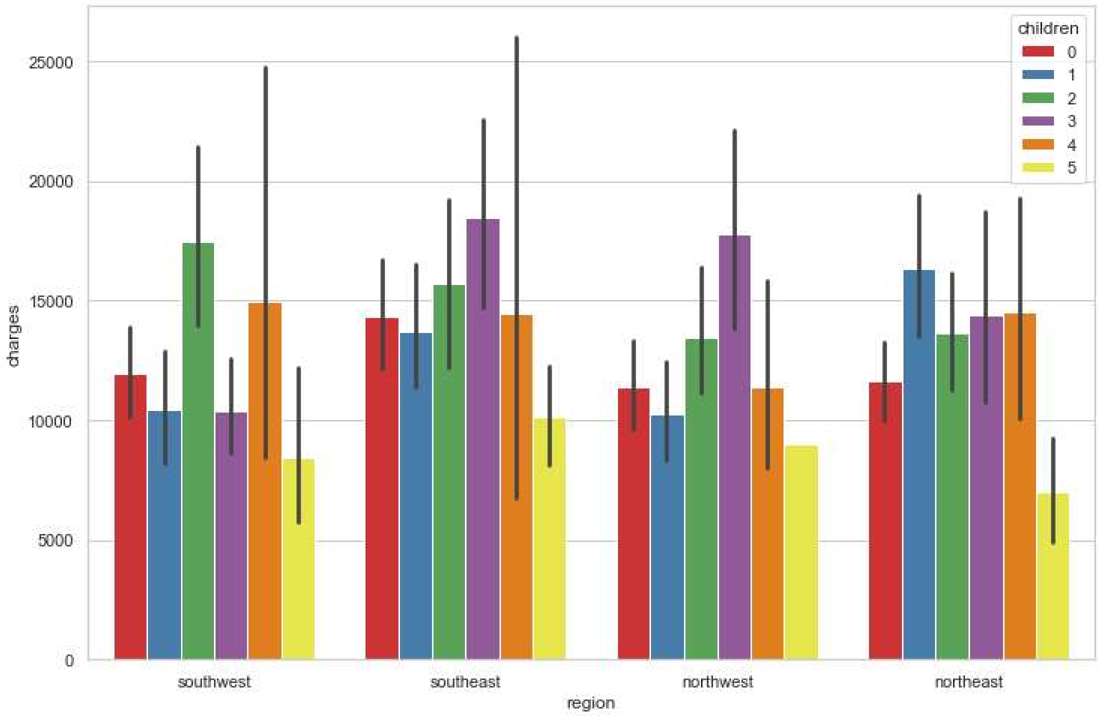
In [12]:
```
f, ax = plt.subplots(1, 1, figsize=(12, 8))
ax = sns.barplot(x='region', y='charges', hue='sex', data=data, palette='cool')
```



In [13]:
```
f, ax = plt.subplots(1,1, figsize=(12,8))
ax = sns.barplot(x = 'region', y = 'charges',hue='smoker', data=data, palette='Reds_r')
```



In [14]:
```
f, ax = plt.subplots(1, 1, figsize=(12, 8))
ax = sns.barplot(x='region', y='charges', hue='children', data=data, palette='Set1')
```

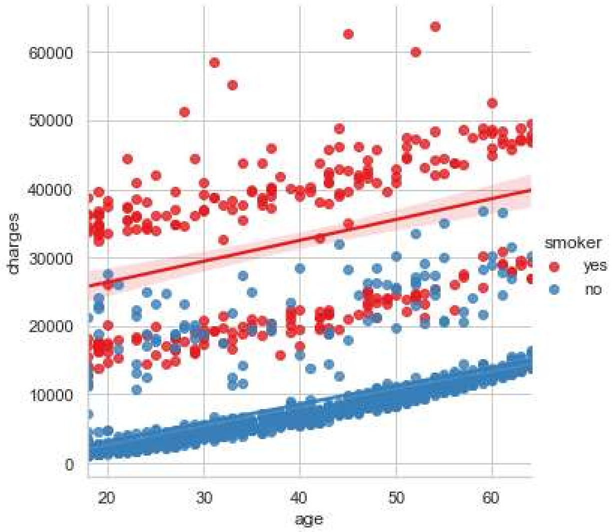


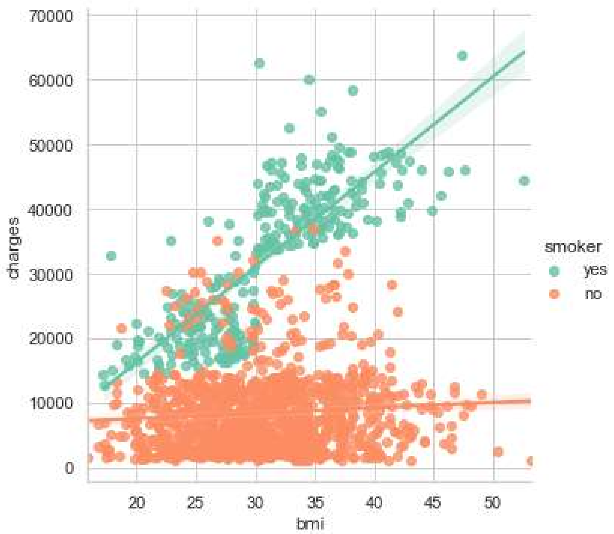
我们可以从这些图表看到，由于吸烟而收取的费用最高的仍然是在东南部，但最低的是在东北部。西南地区的人通常比东北地区的人吸烟

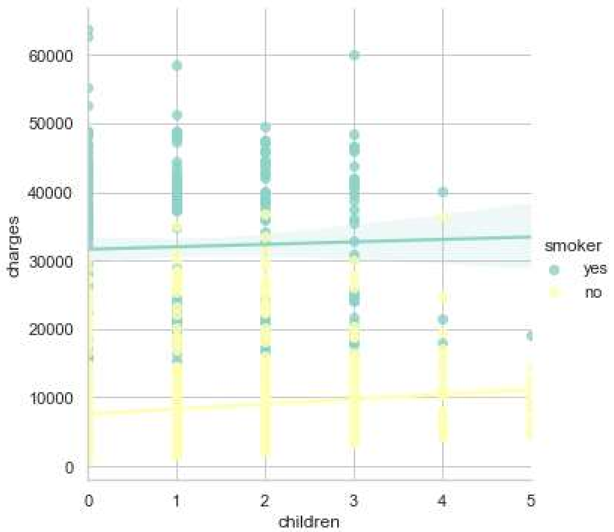
更多，但东北地区的人总体上比西南和西北地区的人有更高的性别收费。总的来说，有孩子的人往往也会有更高的医疗费用。

接下来让我们根据年龄、bmi和儿童吸烟因素来分析医疗费用：

In [15]:
```
ax = sns.lmplot(x = 'age', y = 'charges', data=data, hue='smoker', palette='Set1')
ax = sns.lmplot(x = 'bmi', y = 'charges', data=data, hue='smoker', palette='Set2')
ax = sns.lmplot(x = 'children', y = 'charges', data=data, hue='smoker', palette='Set3')
```

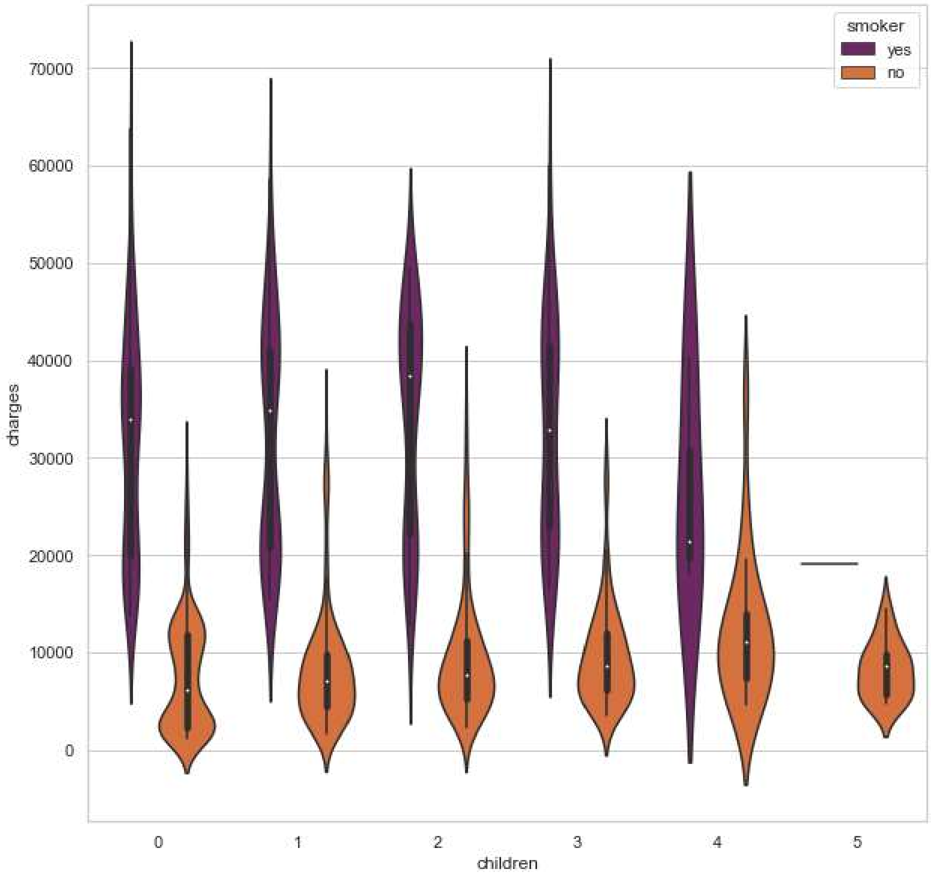






由上图不难分析发现，吸烟对医疗费用的影响最大，尽管医疗费用随着年龄、bmi和儿童的增加而增加。此外，有孩子的人通常吸烟较少，下面的violinplots也显示了这一点。

In [16]:
```
f, ax = plt.subplots(1, 1, figsize=(10, 10))
ax = sns.violinplot(x = 'children', y = 'charges', data=data,orient='v', hue='smoker', palette='inferno')
```



In [17]:
```
##修改object变量数据类型为category

data[['sex', 'smoker', 'region']] = data[['sex', 'smoker', 'region']].astype('category')
data.dtypes
```

Out[17]:
```
age            int64
sex         category
bmi          float64
children       int64
smoker      category
region      category
charges      float64
dtype: object
```

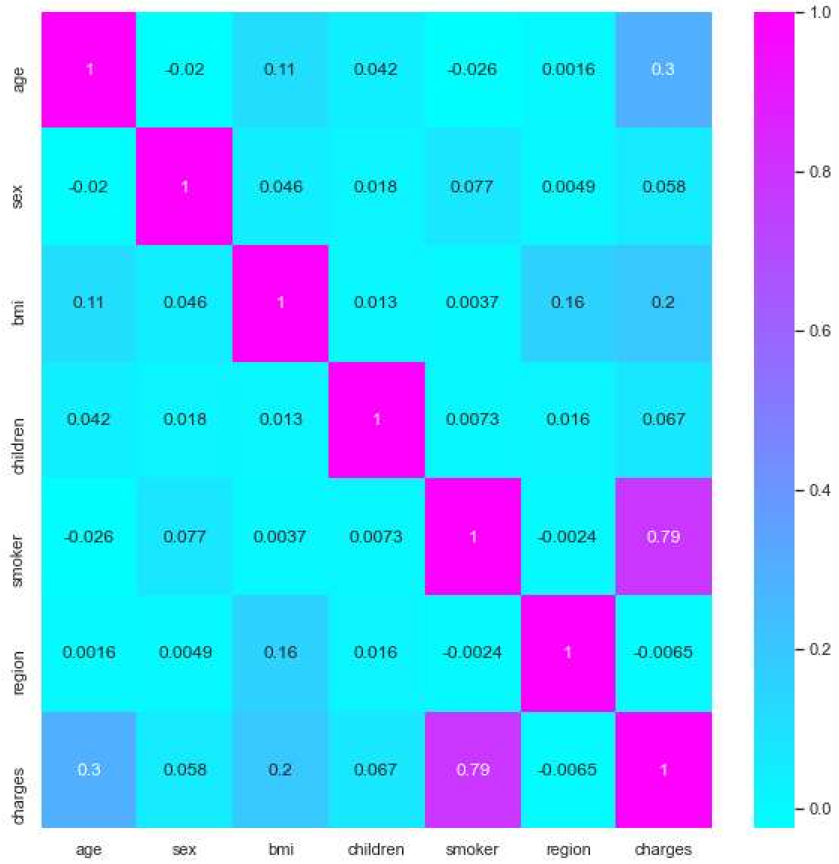
In [18]:
```
##用LabelEncoder将category转化为数值型

from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
label.fit(data.sex.drop_duplicates())
data.sex = label.transform(data.sex)
label.fit(data.smoker.drop_duplicates())
data.smoker = label.transform(data.smoker)
label.fit(data.region.drop_duplicates())
data.region = label.transform(data.region)
data.dtypes
```

Out[18]:
```
age           int64
sex           int32
bmi         float64
children      int64
smoker        int32
region        int32
charges     float64
dtype: object
```

In [19]:
```
##相关图，直观地查看给定数据框（或二维数组）中所有可能的数值变量对之间的相关度量

f, ax = plt.subplots(1, 1, figsize=(10, 10))
ax = sns.heatmap(data.corr(), annot=True, cmap='cool')
```



由此可见，吸烟（smoker）对医疗保险费用的影响最大。

4.回归分析

（1）线性回归

In [20]:
```
from sklearn.model_selection import train_test_split as holdout
from sklearn.linear_model import LinearRegression
from sklearn import metrics
x = data.drop(['charges'], axis = 1)
y = data['charges']
x_train, x_test, y_train, y_test = holdout(x, y, test_size=0.2, random_state=0)
Lin_reg = LinearRegression()
Lin_reg.fit(x_train, y_train)
print(Lin_reg.intercept_)
print(Lin_reg.coef_)
print(Lin_reg.score(x_test, y_test))
```

```
-10658.974155442043
[  244.40254189  -203.81680641   308.01805142   495.56546634
 23771.78167483  -377.96465113]
0.7526726290709553
```

（2）岭回归

In [21]:
```
from sklearn.linear_model import Ridge
Ridge = Ridge(alpha=0.5)
Ridge.fit(x_train, y_train)
print(Ridge.intercept_)
print(Ridge.coef_)
print(Ridge.score(x_test, y_test))
```

```
-10641.314091338594
[   244.34042397  -201.10723643    307.92303881    495.90249348
 23702.74866396  -377.99799347]
0.7526240353110314
```

（3）Lasso回归

In [22]:
```
from sklearn.linear_model import Lasso
Lasso = Lasso(alpha=0.2, fit_intercept=True, precompute=False, max_iter=1000,
              tol=0.0001, warm_start=False, positive=False, random_state=None, selection='cyclic')
Lasso.fit(x_train, y_train)
print(Lasso.intercept_)
print(Lasso.coef_)
print(Lasso.score(x_test, y_test))
```

```
-10658.805921364998
[   244.40242819  -202.95303181    308.0032782     495.42557944
 23770.5047471    -377.78642988]
0.7526729046927889
```

（4）随机森林回归

In [23]:
```
from sklearn.ensemble import RandomForestRegressor as rfr
x = data.drop(['charges'], axis=1)
y = data.charges
Rfr = rfr(n_estimators = 100, criterion = 'squared_error',
                              random_state = 1,
                              n_jobs = -1)
Rfr.fit(x_train,y_train)
x_train_pred = Rfr.predict(x_train)
x_test_pred = Rfr.predict(x_test)

print('MSE train data: %.3f, MSE test data: %.3f' %
      (metrics.mean_squared_error(x_train_pred, y_train),
       metrics.mean_squared_error(x_test_pred, y_test)))
print('R2 train data: %.3f, R2 test data: %.3f' %
      (metrics.r2_score(x_train_pred, y_train),
       metrics.r2_score(x_test_pred, y_test)))
```

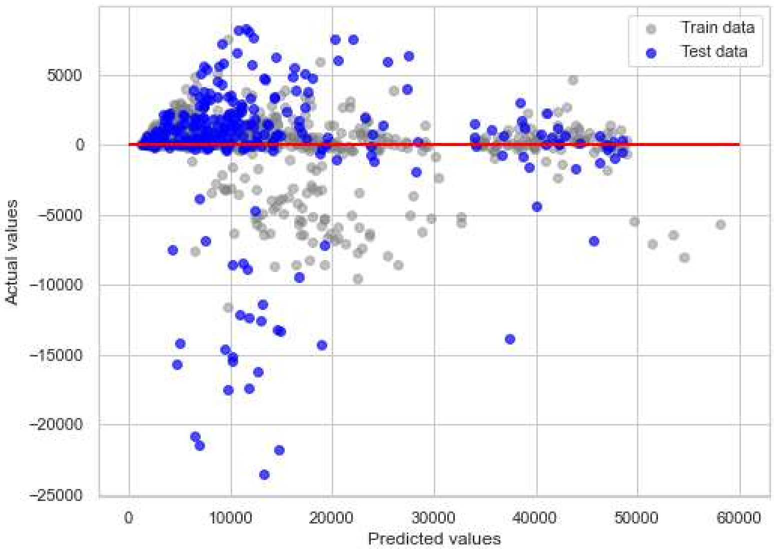
```
MSE train data: 3342536.635, MSE test data: 27301132.995
R2 train data: 0.974, R2 test data: 0.803
```

In [24]:

```
plt.figure(figsize=(8,6))

plt.scatter(x_train_pred, x_train_pred - y_train,
          c = 'gray', marker = 'o', s = 35, alpha = 0.5,
          label = 'Train data')
plt.scatter(x_test_pred, x_test_pred - y_test,
          c = 'blue', marker = 'o', s = 35, alpha = 0.7,
          label = 'Test data')
plt.xlabel('Predicted values')
plt.ylabel('Actual values')
plt.legend(loc = 'upper right')
plt.hlines(y = 0, xmin = 0, xmax = 60000, lw = 2, color = 'red')
```

Out[24]: <matplotlib.collections.LineCollection at 0x1910c718f48>



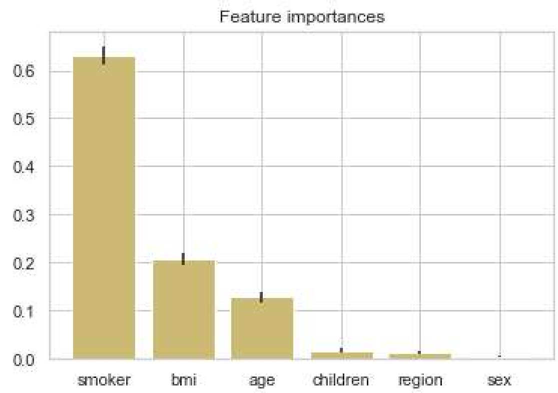
In [25]:
```
print('Feature importance ranking\n\n')
importances = Rfr.feature_importances_
std = np.std([tree.feature_importances_ for tree in Rfr.estimators_],axis=0)
indices = np.argsort(importances)[::-1]
variables = ['age', 'sex', 'bmi', 'children','smoker', 'region']
importance_list = []
for f in range(x.shape[1]):
    variable = variables[indices[f]]
    importance_list.append(variable)
    print("%d.%s(%f)" % (f + 1, variable, importances[indices[f]]))

# Plot the feature importances of the forest
plt.figure()
plt.title("Feature importances")
plt.bar(importance_list, importances[indices],
       color="y", yerr=std[indices], align="center")
```

```
Feature importance ranking


1.smoker(0.629024)
2.bmi(0.206035)
3.age(0.128738)
4.children(0.017091)
5.region(0.013517)
6.sex(0.005595)
```

Out[25]: <BarContainer object of 6 artists>



（5）多项式回归

In [26]:
```
from sklearn.preprocessing import PolynomialFeatures
x = data.drop(['charges', 'sex', 'region'], axis = 1)
y = data.charges
pol = PolynomialFeatures (degree = 2)
x_pol = pol.fit_transform(x)
x_train, x_test, y_train, y_test = holdout(x_pol, y, test_size=0.2, random_state=0)
Pol_reg = LinearRegression()
Pol_reg.fit(x_train, y_train)
y_train_pred = Pol_reg.predict(x_train)
y_test_pred = Pol_reg.predict(x_test)
print(Pol_reg.intercept_)
print(Pol_reg.coef_)
print(Pol_reg.score(x_test, y_test))
```

```
-4154.172427989844
[ 0.00000000e+00  4.58324545e-01  4.03405523e+02  1.40260553e+03
 -1.12260916e+04  3.01311573e+00  3.24610178e-01 -5.35321284e-01
  2.51089226e+01 -6.58358646e+00 -8.22584809e+00  1.48200077e+03
 -1.41284021e+02 -3.75316527e+01 -1.12260916e+04]
0.8357482955959796
```

In [27]:
```
##Evaluating the performance of the algorithm

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_test_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_test_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
Mean Absolute Error: 3175.8186596347696
Mean Squared Error: 27631993.10614759
Root Mean Squared Error: 5256.614224588637
```

In [28]:
```
##Predicting the charges

y_test_pred = Pol_reg.predict(x_test)

##Comparing the actual output values with the predicted values

df = pd.DataFrame({'Actual': y_test, 'Predicted': y_test_pred})
df
```

Out[28]:

	Actual	Predicted
1248	1633.96180	2687.454432
610	8547.69130	10135.037321
393	9290.13950	10754.902954
503	32548.34050	26162.873761
198	9644.25250	9141.656812
...	...	...
809	3309.79260	5013.664929
726	6664.68595	8460.233673
938	2304.00220	4822.210691
474	25382.29700	29294.007454
1084	15019.76005	15921.876412

268 rows × 2 columns

5.结论

吸烟是影响医疗费用的最大因素，其次是bmi和年龄;同时多项式回归被证明是最好的模型。