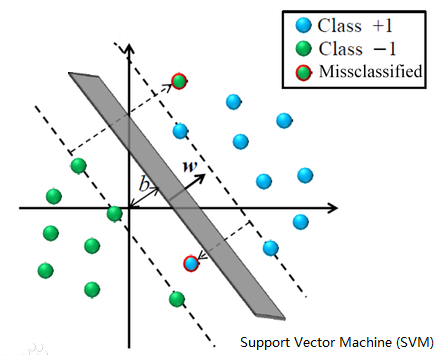
# 程序实现

## 模型选择

我们要根据数据判断出地产是否可以买，只有买和不买，二分的svm比较适合我们的数据



核函数我们选择了——径向基核函数（RBF）

1963年Davis提出高维空间的多变量插值理论。径向基函数技术则是20世纪80年代后期，Powell在解决“多变量有限点严格（精确）插值问题”时引人的，目前径向基函数已成为数值分析研究中的一个重要领域。

我们变量比较多，并且非线性，所以使用的径向基函数

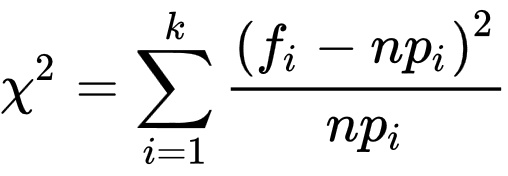
## 特征选择

**方差分析**

方差分析(Analysis of Variance，简称ANOVA)，又称“变异数分析”，是[R.A.Fisher](https://baike.baidu.com/item/R.A.Fisher/8606035)发明的，用于两个及两个以上[样本](https://baike.baidu.com/item/%E6%A0%B7%E6%9C%AC/5712132)均数差别的[显著性检验](https://baike.baidu.com/item/%E6%98%BE%E8%91%97%E6%80%A7%E6%A3%80%E9%AA%8C/8788495)。 由于各种因素的影响，研究所得的[数据](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE/33305)呈现波动状。造成波动的原因可分成两类，一是不可控的随机因素，另一是研究中施加的对结果形成影响的[可控因素](https://baike.baidu.com/item/%E5%8F%AF%E6%8E%A7%E5%9B%A0%E7%B4%A0/19223913)。

**卡方检验**

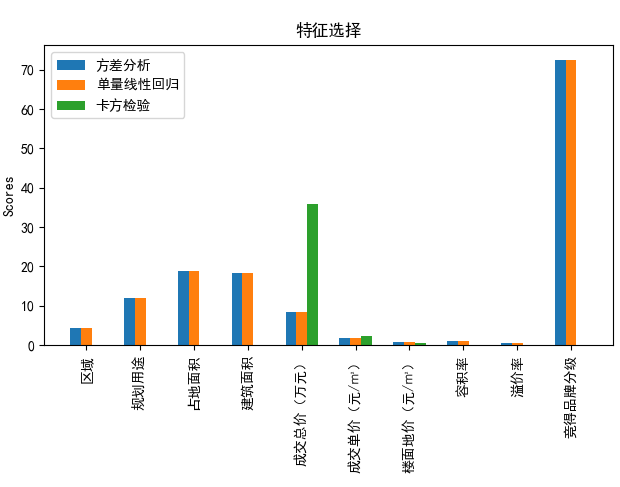
卡方检验就是统计[样本](https://baike.baidu.com/item/%E6%A0%B7%E6%9C%AC/5712132)的实际观测值与理论推断值之间的偏离程度，实际观测值与理论推断值之间的偏离程度就决定卡方值的大小，如果卡方值越大，二者偏差程度越大；反之，二者偏差越小；若两个值完全相等时，卡方值就为0，表明理论值完全符合。



**单量线性回归**

线性回归是利用数理统计中回归分析，来确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法，运用十分广泛。其表达形式为y = w'x+e，e为误差服从均值为0的正态分布。

计算相关性



**源码**

我们根据

from sklearn.feature\_selection import SelectKBest, f\_classif, f\_regression, chi2

from modules.dataset import X, y, columns

import matplotlib.pyplot as plt

import numpy as np

d = {}

# f\_classif

selector = SelectKBest(score\_func=f\_classif, k=3)

selector.fit(X, y)

d["f\_classif"] = selector.scores\_

print("f\_classif")

print('pvalues\_:',selector.pvalues\_)

print('selected index:',selector.get\_support(True))

# f\_regression

selector = SelectKBest(score\_func=f\_regression, k=3)

selector.fit(X, y)

d["f\_regression"] = selector.scores\_

print('pvalues\_:',selector.pvalues\_)

print('selected index:',selector.get\_support(True))

# chi

selector = SelectKBest(score\_func=chi2, k=3)

selector.fit(X, y)

d["chi2"] = selector.scores\_

print("chi2")

print('pvalues\_:',selector.pvalues\_)

print('selected index:',selector.get\_support(True))

plt.rcParams['font.sans-serif'] = ['SimHei']  # 中文字体设置

plt.rcParams['axes.unicode\_minus'] = False

fig, ax = plt.subplots()

# Add some text for labels, title and custom x-axis tick labels, etc.

ax.set\_ylabel('Scores')

ax.set\_title('特征选择')

x = np.arange(len(columns))  # the label locations

ax.set\_xticks(x)

ax.set\_xticklabels(columns)

for tick in ax.get\_xticklabels():

  tick.set\_rotation(90)

width = 0.20  # the width of the bars

ax.bar(x - width, d["f\_classif"], width, label='方差分析')

ax.bar(x, d["f\_regression"], width, label='单量线性回归')

ax.bar(x + width, d["chi2"]/10000, width, label='卡方检验')

ax.legend()

fig.tight\_layout()

plt.show()

**模块**

数据模块专门分离出来

import pandas as pd

df = pd.read\_excel("estate.xlsx", sheet\_name="粗加工")

df.drop(0, inplace = True)

columns = [

  # '宗地编号',

  # '成交时间',

  '区域',

  '规划用途',

  '占地面积',

  '建筑面积',

  '成交总价（万元）',

  '成交单价（元/㎡）',

  '楼面地价（元/㎡）',

  '容积率',

  '溢价率',

  '竞得品牌分级',

  # '项目验证（标签）'

]

X = df.loc[:, columns].values

y = df.loc[:,  '项目验证（标签）'].values

根据图表

方差分析和单量线性回归的表现看，占地面积，建筑面积，竞得品牌分级相关性比较大

卡方卡方检验中成交总价相关性比较大，其次是成交单价和楼面地价

实际测试一下准确率：

特征选择让准确率得到了比较明显的提高，均值提高一个百分点，众数更是提高9个百分点。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 全部 | 选定特征 | 削峰 |
|  |  |  |  |
| 平均 | 0.793333 | 0.809524 | 0.797143 |
| 标准误差 | 0.011104 | 0.011464 | 0.011204 |
| 中位数 | 0.785714 | 0.809524 | 0.809524 |
| 众数 | 0.761905 | 0.857143 | 0.714286 |
| 标准差 | 0.078518 | 0.081064 | 0.079222 |
| 方差 | 0.006165 | 0.006571 | 0.006276 |
| 峰度 | -0.34656 | -0.52846 | 0.142382 |
| 偏度 | 0.174745 | -0.07757 | 0.016909 |
| 区域 | 0.333333 | 0.333333 | 0.380952 |
| 最小值 | 0.619048 | 0.619048 | 0.571429 |
| 最大值 | 0.952381 | 0.952381 | 0.952381 |
| 求和 | 39.66667 | 40.47619 | 39.85714 |
| 观测数 | 50 | 50 | 50 |
| 最大(1) | 0.952381 | 0.952381 | 0.952381 |
| 最小(1) | 0.619048 | 0.619048 | 0.571429 |
| 置信度(95.0%) | 0.022314 | 0.023038 | 0.022515 |

**源码**

换不同的特征，对比正确率，输出到xlsx上，之后还需要用到dmd的数据描述

import modules.svm as svm

import pandas as pdw

import time

dfw = pdw.DataFrame({"id":[], "全部":[], "选定特征":[], "削峰":[]})

print(r"{0:-^50}".format("开始执行"))

start = time.perf\_counter()

num = 50

for i in range(1, num + 1):

  whole = svm.run(i)

  selected = svm.run(i, feagures = [2,3,4,9])

  flat = svm.run(i, feagures = [2,3,4])

  row = { "id": whole["seed"], "全部": whole["accuracy"], "选定特征": selected["accuracy"], "削峰": flat["accuracy"] }

  dfw.loc[i] = row

  now = time.perf\_counter() - start

  per = (i/num)\*100

  star = "\*" \* int(per)

  dot = "." \* (num - int(per))

  print("\r{:.2f}%[{}->{}]{:.2f}s".format(per, star, dot, now), end='')

dfw.to\_excel("out/feagure.xlsx", sheet\_name='明细', index=False)

print("\n"+"{0:-^50}".format("执行结束"))

print("{0:-^50}".format("feagure.xlsx生成"))

**模块**

因为svm的操作很多，所以专门做了svm的模块多次调用

from sklearn.model\_selection import train\_test\_split

from sklearn.pipeline import make\_pipeline

from sklearn.preprocessing import StandardScaler, MinMaxScaler

from sklearn.metrics import accuracy\_score,precision\_score, recall\_score

from sklearn.svm import SVC

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

df = pd.read\_excel("estate.xlsx", sheet\_name="粗加工") # 读取xlsx数据

df.drop(0, inplace = True) # 标题有两个，需要去掉一个

def run(seed = 0, C = 1, feagures = []):

  """运行svm

  Parameters

  ----------

  seed: 随机种子

  C: 学习力度，过小会“欠学习”，过大会“过学习”

  feagues: 特征选择

  """

  pipe = make\_pipeline(

    StandardScaler(),

    SVC(random\_state = 0, C = C)

  )

  columns = [

    # '宗地编号',

    # '成交时间',

    '区域',

    '规划用途',

    '占地面积',

    '建筑面积',

    '成交总价（万元）',

    '成交单价（元/㎡）',

    '楼面地价（元/㎡）',

    '容积率',

    '溢价率',

    '竞得品牌分级',

    # '项目验证（标签）'

  ]

  if feagures and len(feagures) > 0:

    arr = []

    for i in feagures:

      arr.append(columns[i])

    columns = arr

  X = df.loc[:, columns].values

  y = df.loc[:,  '项目验证（标签）'].values

  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state = seed)

  pipe.fit(X\_train, y\_train)

  predictions = pipe.predict(X\_test)

  return {

    "seed": seed,

    "accuracy": accuracy\_score(y\_test, predictions),

    "precision": precision\_score(y\_test, predictions),

    "recall": recall\_score(y\_test, predictions)

  }

def predict(C = 1, feagures = []):

  """预测

  Parameters

  ----------

  C: 学习力度，过小会“欠学习”，过大会“过学习”

  feagues: 特征选择

  """

  pipe = make\_pipeline(

    StandardScaler(),

    SVC(random\_state = 0, C = C)

  )

  columns = [

    # '宗地编号',

    # '成交时间',

    '区域',

    '规划用途',

    '占地面积',

    '建筑面积',

    '成交总价（万元）',

    '成交单价（元/㎡）',

    '楼面地价（元/㎡）',

    '容积率',

    '溢价率',

    '竞得品牌分级',

    # '项目验证（标签）'

  ]

  if feagures and len(feagures) > 0:

    arr = []

    for i in feagures:

      arr.append(columns[i])

    columns = arr

  X = df.loc[:, columns].values

  y = df.loc[:,  '项目验证（标签）'].values

  pipe.fit(X, y)

  reality = pd.read\_excel("estate.xlsx", sheet\_name="2020粗加工")

  reality.drop(0, inplace = True)

  real\_X = reality.loc[:, columns].values

  real\_y = reality.loc[:,  '项目验证（标签）'].values

  predictions = pipe.predict(real\_X)

  return {

    "accuracy": accuracy\_score(real\_y, predictions),

    "precision": precision\_score(real\_y, predictions),

    "recall": recall\_score(real\_y, predictions),

    "origin": reality.loc[:].values,

    "predictions": predictions

  }

## 参数调整

调整惩罚系数，学习力度，过小会“欠学习”，过大会“过学习”

我们先从0.1到20之间取一下数据

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0.1 | 0.5 | 1 | 5 | 7 | 10 | 15 | 20 |
| 平均 | 0.705714 | 0.810476 | 0.809524 | 0.831429 | 0.827619 | 0.818095 | 0.812381 | 0.814286 |
| 标准误差 | 0.012668 | 0.010755 | 0.011464 | 0.010731 | 0.010617 | 0.009782 | 0.010128 | 0.009835 |
| 中位数 | 0.666667 | 0.809524 | 0.809524 | 0.833333 | 0.833333 | 0.809524 | 0.809524 | 0.809524 |
| 众数 | 0.666667 | 0.809524 | 0.857143 | 0.761905 | 0.761905 | 0.857143 | 0.857143 | 0.857143 |
| 标准差 | 0.089574 | 0.076051 | 0.081064 | 0.07588 | 0.075071 | 0.069167 | 0.071613 | 0.069541 |
| 方差 | 0.008024 | 0.005784 | 0.006571 | 0.005758 | 0.005636 | 0.004784 | 0.005128 | 0.004836 |
| 峰度 | 0.5248 | -0.14136 | -0.52846 | -1.13859 | -1.17987 | -1.21911 | -1.10875 | -0.52055 |
| 偏度 | 0.711985 | -0.2531 | -0.07757 | 0.138883 | 0.083705 | -0.07712 | -0.1435 | -0.13925 |
| 区域 | 0.428571 | 0.333333 | 0.333333 | 0.238095 | 0.238095 | 0.238095 | 0.285714 | 0.285714 |
| 最小值 | 0.52381 | 0.619048 | 0.619048 | 0.714286 | 0.714286 | 0.714286 | 0.666667 | 0.666667 |
| 最大值 | 0.952381 | 0.952381 | 0.952381 | 0.952381 | 0.952381 | 0.952381 | 0.952381 | 0.952381 |
| 求和 | 35.28571 | 40.52381 | 40.47619 | 41.57143 | 41.38095 | 40.90476 | 40.61905 | 40.71429 |
| 观测数 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 |
| 最大(1) | 0.952381 | 0.952381 | 0.952381 | 0.952381 | 0.952381 | 0.952381 | 0.952381 | 0.952381 |
| 最小(1) | 0.52381 | 0.619048 | 0.619048 | 0.714286 | 0.714286 | 0.714286 | 0.666667 | 0.666667 |
| 置信度(95.0%) | 0.025457 | 0.021613 | 0.023038 | 0.021565 | 0.021335 | 0.019657 | 0.020352 | 0.019763 |

根据数据，查看，惩罚系数设定在10的时候，平均值，众数，最小值比较高，标准差，方差比较低，是比较合适的值

**源码**

根据不同的惩罚系数检验准确率，输出到xlsx

import modules.svm as svm

import pandas as pdw

import time

dfw = pdw.DataFrame({"id":[], "0.1":[], "0.5":[], "1":[], "5":[], "7":[], "10":[], "15":[], "20":[]})

print(r"{0:-^50}".format("开始执行"))

start = time.perf\_counter()

num = 50

feagures = [2,3,4,5,6,9]

for i in range(1, num + 1):

  res01 = svm.run(i, 0.1, feagures)

  res05 = svm.run(i, 0.5, feagures)

  res1 = svm.run(i, 1, feagures)

  res2 = svm.run(i, 2, feagures)

  res5 = svm.run(i, 5, feagures)

  res7 = svm.run(i, 7, feagures)

  res10 = svm.run(i, 10, feagures)

  res15 = svm.run(i, 15, feagures)

  res20 = svm.run(i, 20, feagures)

  row = { "id": res1["seed"], "0.1": res01["accuracy"], "0.5": res05["accuracy"],

    "1": res1["accuracy"], "2": res2["accuracy"], "5": res5["accuracy"],

    "7": res7["accuracy"],

    "10": res10["accuracy"],

    "15": res15["accuracy"],

    "20": res20["accuracy"]

  }

  dfw.loc[i] = row

  now = time.perf\_counter() - start

  per = i/num

  star = "\*" \* int(per \* num)

  dot = "." \* (num - int(per \* num))

  print("\r{:.2f}%[{}->{}]{:.2f}s".format(per \* 100, star, dot, now), end='')

dfw.to\_excel("out/c.xlsx", sheet\_name='明细', index=False)

print("\n"+"{0:-^50}".format("执行结束"))

print("{0:-^50}".format("c.xlsx生成"))

调整gamma，gamma值代表的拟合程度，简单来说就是弯曲程度，过高会过拟合

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0.1 | 0.5 | 1 | 5 | 7 | 10 | 15 | 20 |
|  |  |  |  |  |  |  |  |  |
| 平均 | 0.851429 | 0.822857 | 0.820952 | 0.840952 | 0.846667 | 0.847619 | 0.848571 | 0.830476 |
| 标准误差 | 0.009191 | 0.009528 | 0.009864 | 0.009188 | 0.009648 | 0.010181 | 0.010153 | 0.010286 |
| 中位数 | 0.857143 | 0.857143 | 0.833333 | 0.857143 | 0.857143 | 0.857143 | 0.857143 | 0.833333 |
| 众数 | 0.857143 | 0.857143 | 0.857143 | 0.857143 | 0.857143 | 0.857143 | 0.857143 | 0.857143 |
| 标准差 | 0.064994 | 0.067371 | 0.069747 | 0.064972 | 0.068224 | 0.071993 | 0.071794 | 0.072735 |
| 方差 | 0.004224 | 0.004539 | 0.004865 | 0.004221 | 0.004655 | 0.005183 | 0.005154 | 0.00529 |
| 峰度 | -0.48339 | -0.44337 | -0.20783 | -0.92474 | -0.95382 | -0.94136 | -0.8969 | -0.71713 |
| 偏度 | -0.42686 | -0.65673 | -0.39495 | -0.10035 | -0.20302 | -0.34696 | -0.38809 | -0.18589 |
| 区域 | 0.238095 | 0.238095 | 0.285714 | 0.238095 | 0.238095 | 0.238095 | 0.238095 | 0.285714 |
| 最小值 | 0.714286 | 0.666667 | 0.666667 | 0.714286 | 0.714286 | 0.714286 | 0.714286 | 0.666667 |
| 最大值 | 0.952381 | 0.904762 | 0.952381 | 0.952381 | 0.952381 | 0.952381 | 0.952381 | 0.952381 |
| 求和 | 42.57143 | 41.14286 | 41.04762 | 42.04762 | 42.33333 | 42.38095 | 42.42857 | 41.52381 |
| 观测数 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 |
| 最大(1) | 0.952381 | 0.904762 | 0.952381 | 0.952381 | 0.952381 | 0.952381 | 0.952381 | 0.952381 |
| 最小(1) | 0.714286 | 0.666667 | 0.666667 | 0.714286 | 0.714286 | 0.714286 | 0.714286 | 0.666667 |
| 置信度(95.0%) | 0.018471 | 0.019147 | 0.019822 | 0.018465 | 0.019389 | 0.02046 | 0.020404 | 0.020671 |

虽然比较细微的差别，最终我们选定了15

## 检验：

用今年的数据测试，准确率有所降低

大概是因为疫情造成的大企业没有拿地

|  |  |  |
| --- | --- | --- |
| **准确度** | **精确度** | **召回率** |
| 0.684211 | 0.69697 | 0.92 |

准确度接近70%

**源码**

用新的数据预测，并输出所有数据的预测值以及准确率

import modules.svm as svm

import pandas as pd

import numpy as np

feagures = [2, 3, 4, 9]

res = svm.predict(10, feagures)

df = pd.read\_excel("estate.xlsx", sheet\_name="2020粗加工")

df = df.drop(df.columns[-3:], axis=1)

df["预测"] = np.insert(res["predictions"], 0, -1, axis=0)

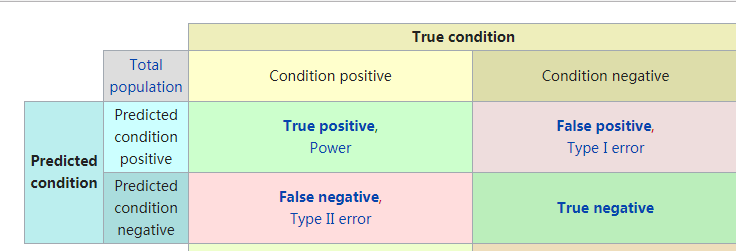
writer = pd.ExcelWriter("./out/reality.xlsx") # pylint: disable=abstract-class-instantiated

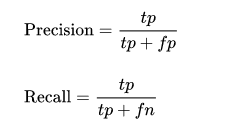
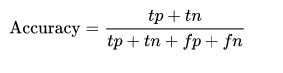
df.to\_excel(writer, sheet\_name="数据", index=False)

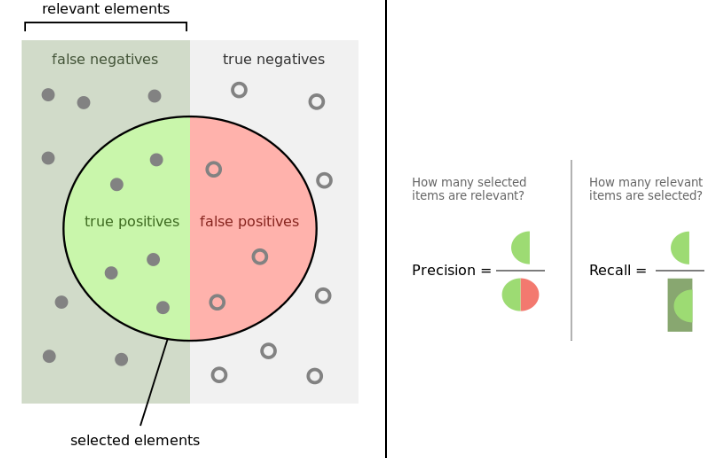
factor = pd.DataFrame({"准确度":[res["accuracy"]], "精确度":[res["precision"]], "召回率":[res["recall"]]},)

factor.to\_excel(writer, sheet\_name="精确", index=False)

writer.save()







## 源码下载

Github：<https://github.com/thales-study/estate.git>

Gitee：<https://gitee.com/thales-jiang/estate.git>

下载到本地可以直接运行

语言：python3.x

库：sklearn matplotlib numpy pandas