**利用机器学习预测房价**

目录

[第一部分 介绍 1](#_Toc29729487)

[第二部分 问题分析 4](#_Toc29729488)

[第三部分 模型应用与分析 8](#_Toc29729489)

[第四部分 总结 14](#_Toc29729490)

第一部分 介绍

在本次实验项目中，我们主要以kaggle “House Prices: Advanced Regression Techniques”为例，来探究普通机器学习方法（如线性回归、SVM）不能很好地实现我们的需求的时候，如何综合各种机器学习方法的优势来帮助我们解决问题，从而构建工程化的应用。通过Kaggle“Horse Prices”的数据集与场景分析，在本次实验中我们主要采用回归方法，如：Linear least squares with l2 regularization.，Linear Model trained with L1 prior as regularizer (aka the Lasso)、Epsilon-Support Vector Regression，LightGBM regressor方法。通过具体的代码实践，我们发现这些方法各有优劣，有的比较耗费算力，有的比较占用内存空间，而有的效果表现在某些情况下并不理想。而生产实践中，我们往往把稳定性看的很重要。在这里我们通过集成学习的方法，来综合各种学习效果，从多种学习的方法论上，解决单一学习方法的不足，从而获得比较好的应用效果。同时在机器学习的具体应用中，往往存在调参比较难的情况。往往机器学习工程师，在这个过程中变得比较机械地试验数据。耗费时间跟精力，我们在这里使用了交叉验证。通过认为给定参数范围来选择其中拟合效果最佳的输出值。

在实验项目中，我们采用MLXTEND的StackingCVReressor方法，这个方法集合中StackingRegressor集成学习元回归器与Cross Values交叉验证的功能。我们分别对两种方法来做介绍。

**集成学习**

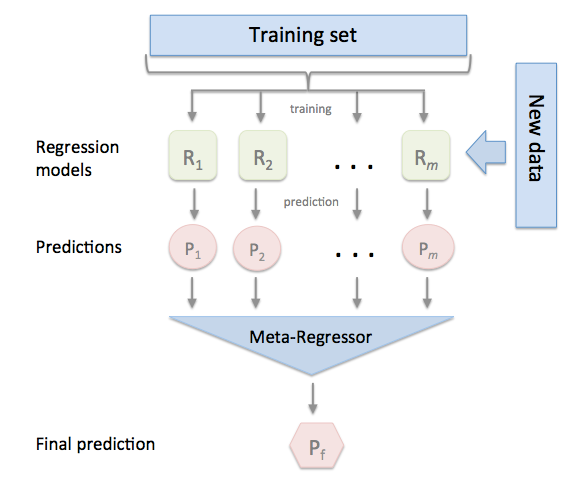
集成学习种stacking回归器是一种通过元回归器组成多个回归模型的集成学习技术。每个独立的底层回归模型再训练是都要使用这个训练数据集。再集成学习种我们是把底层回归器作为stacking回归器的输入。Stacking回归器通过拟合底层回归器来组合各种模型。在之后的计算结果种，我们采用平均法的策略为作为返回值。

对每一个学习器会有一个权重w, 则最终预测为：

其中是底层学习器的权重，有

=1

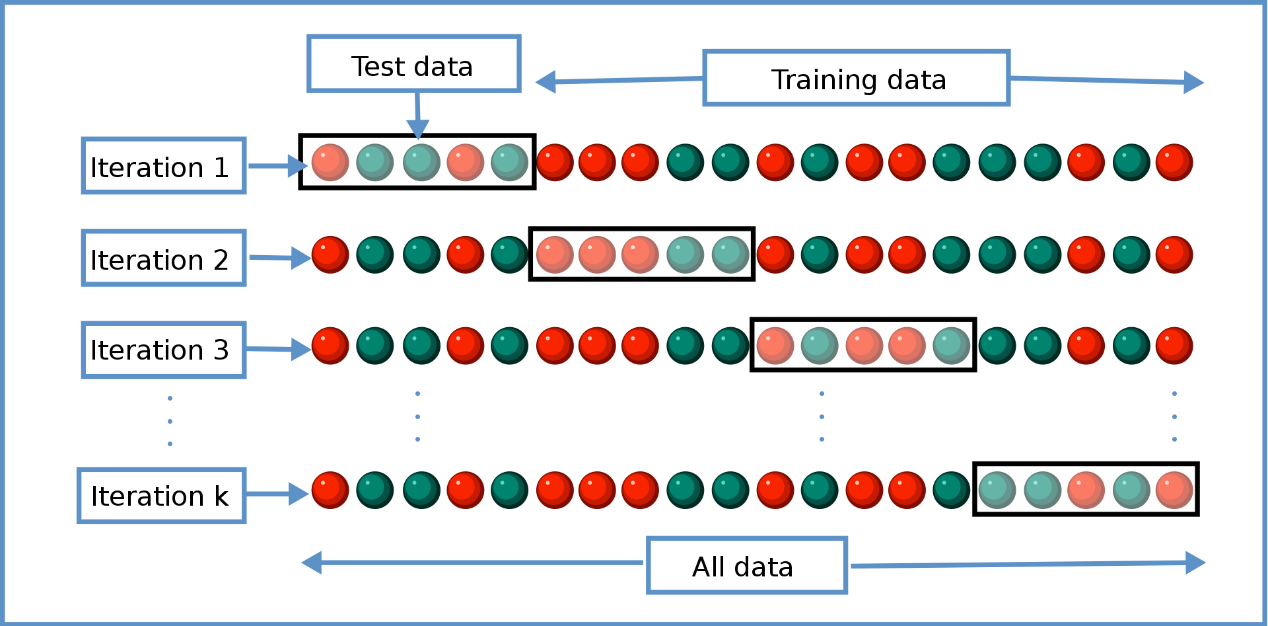
其基本原理如下图所示：



**交叉验证**

交叉验证是在在机器学习建立模型和验证模型参数下使用的，用于评估学习器的表现。这里我们用到的是K折交叉验证（K-Folder Cross Validation）。K折交叉验证将数据集D按某种正态分布随机划分为K个数据量相同的互斥自己。即

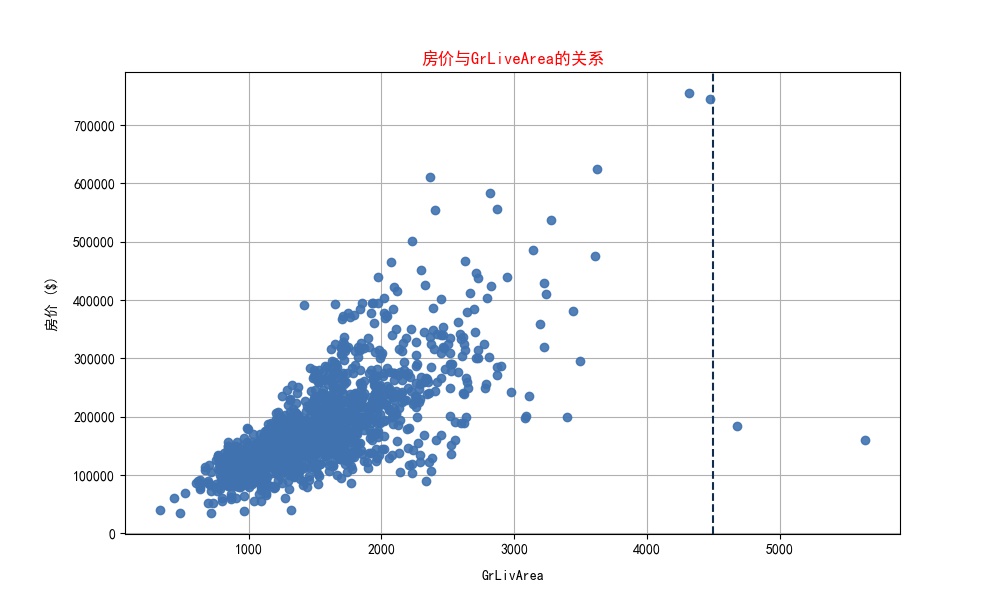
每次选择K-1份作为训练数据集，另外一份作为测试数据集。导入到学习器模型重，选择损失函数评估最优的模型与参数。



第二部分 问题分析

在我们遇到的问题中，这里主要透过kaggle 中提供的数据集与数据说明来深度分析我们所遇到的问题。首先我们有71个特征变量，1个目标变量。房价（即SalePrice）是我们要做的预测值。

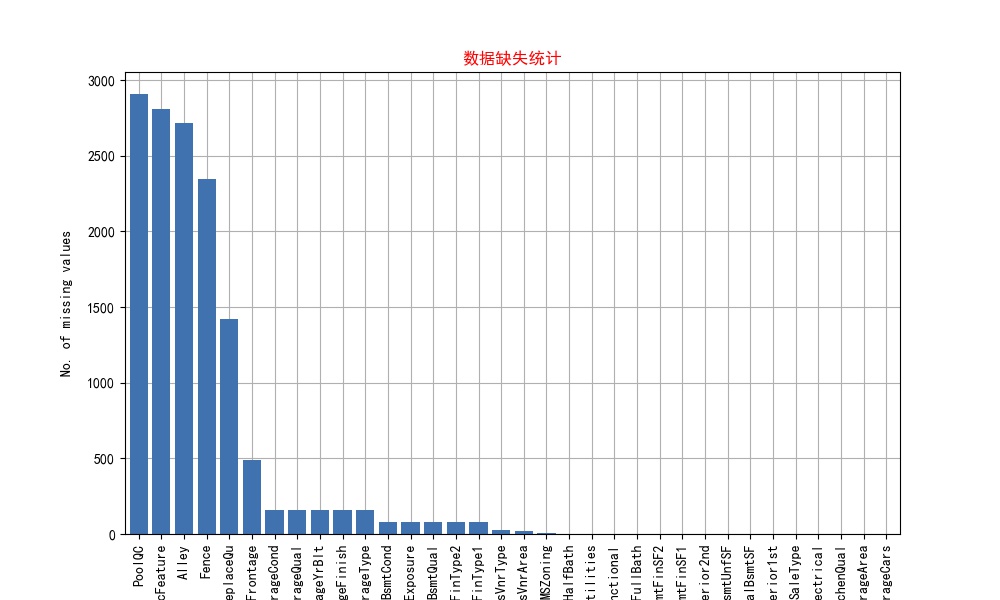
首先我们用地面以上居住面积平方英尺（在数据集中为GrLivArea数据节点），发现其与SalePrice的变化是这样的。



通过图片所展示其线性特征并不明显。很难用常规的回归方法来解决此类问题。

|  |  |  |  |
| --- | --- | --- | --- |
| 方法 | 介绍 | 优点 | 缺点 |
| Ridge | Linear least squares with l2 regularization | 运行速度快 | 结果不太准 |
| Lasso | Linear Model trained with L1 prior as regularizer (aka the Lasso) | 结果比较良好 | 对参数要求比较高 |
| SVR | Epsilon-Support Vector Regression | 运行结果相对较好 | 比较耗算力 |
| LGBMRegressor | LightGBM regressor | 运行结果优秀 | 运行时间需要很长 |

在目前的训练数据集与测试数据集中我们发现也存在了很多缺失值。为了能否导入到机器学习的方程中，我们需要对这些缺失值进行处理。



采取数据处理的策略为：

1、对于缺失数据比较多的，用None来填充。如PoolQC数据点。

2、对于数据缺失相对比较少的用0来填充

3、特殊含义的数据如Neighborhood， 用出现最多的数值来填充。

4、其余普通字段用中位数填充。

特征工程

基本策略为：

1、对相同意义的特征数据进行合并。如新建TotalSF（住房总面积），TotalPorchSF（门廊面积）， TotalBath（浴缸数量）

有：

TotalSF = GrLivArea + TotalBsmtSF

TotalPorchSF = OpenPorchSF + EnclosedPorch + 3SsnPorch + ScreenPorch

TotalBath = FullBath + BsmtFullBath + 0.5 \* ("BsmtHalfBath + HalfBath)

2、对 MSSubClass 与 YrSold 进行category 处理。在导入计算方法的时候，能够自动对标签型数据转化为数值型数据。

3、对周期性数据正交化

X["SinMoSold"] = np.sin(2 \* np.pi \* X["MoSold"] / 12)  
X["CosMoSold"] = np.cos(2 \* np.pi \* X["MoSold"] / 12)

4、数据预处理

skew = X.skew(numeric\_only=True).abs()  
cols = skew[skew > 1].index  
for col in cols:  
 X[col] = boxcox1p(X[col], boxcox\_normmax(X[col] + 1))  
  
cols = X.select\_dtypes(np.number).columns  
X[cols] = RobustScaler().fit\_transform(X[cols])  
  
X = pd.get\_dummies(X)  
  
X\_train = X.loc[train.index]  
X\_test = X.loc[test.index]

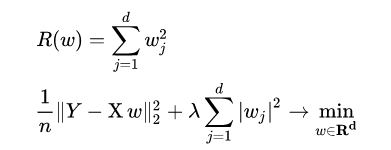
第三部分 模型应用与分析

因为我们的数据量相对较少，所以需要交叉验证来不断导入到模型与分析才能获得比较接近真实水平的值。在这里我们用sklearn的RandomizedSearchCV方法来做数据的切割。其基本原理为，将训练数据集采用统计分布的方法进行采用，随机得到我们要的训练数据与测试数据。并且不断导入到模型中去，选择其中模型所得数据结果与真实结果相差最少的模型结果。

def random\_search(model, grid, n\_iter=100):  
 kf = KFold(n\_splits=5, random\_state=0, shuffle=True)  
 rmse = lambda y, y\_pred: np.sqrt(mean\_squared\_error(y, y\_pred))  
 scorer = make\_scorer(rmse, greater\_is\_better=False)  
 n\_jobs = max(cpu\_count() - 2, 1)  
 search = RandomizedSearchCV(model, grid, n\_iter, scorer, n\_jobs=n\_jobs, cv=kf, random\_state=0, verbose=True)  
 return search.fit(X\_train, y\_train)

1、采用Ridge方法

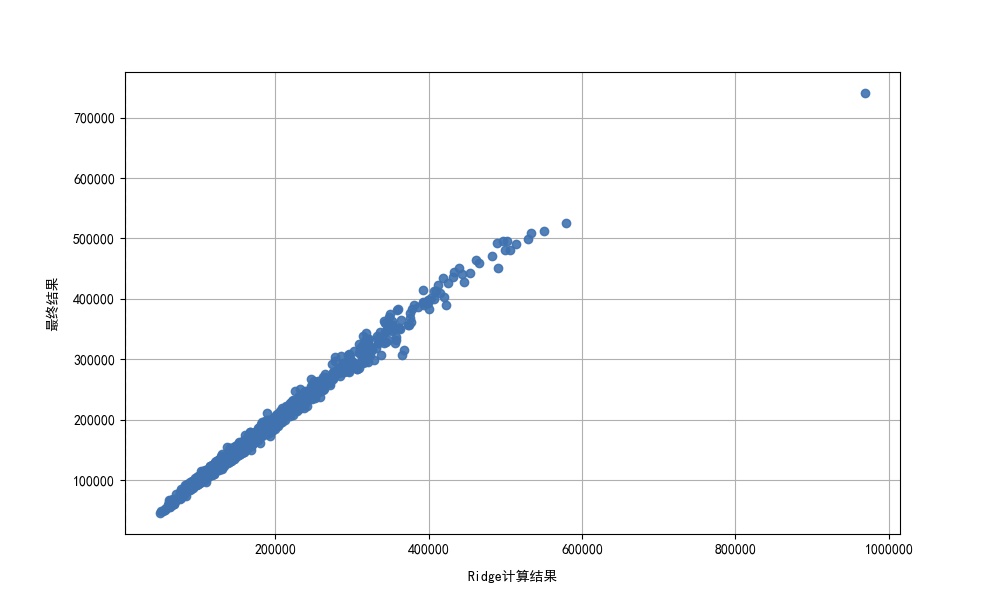
方法介绍:



其中：

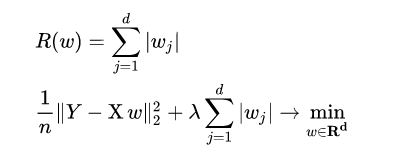


表现：

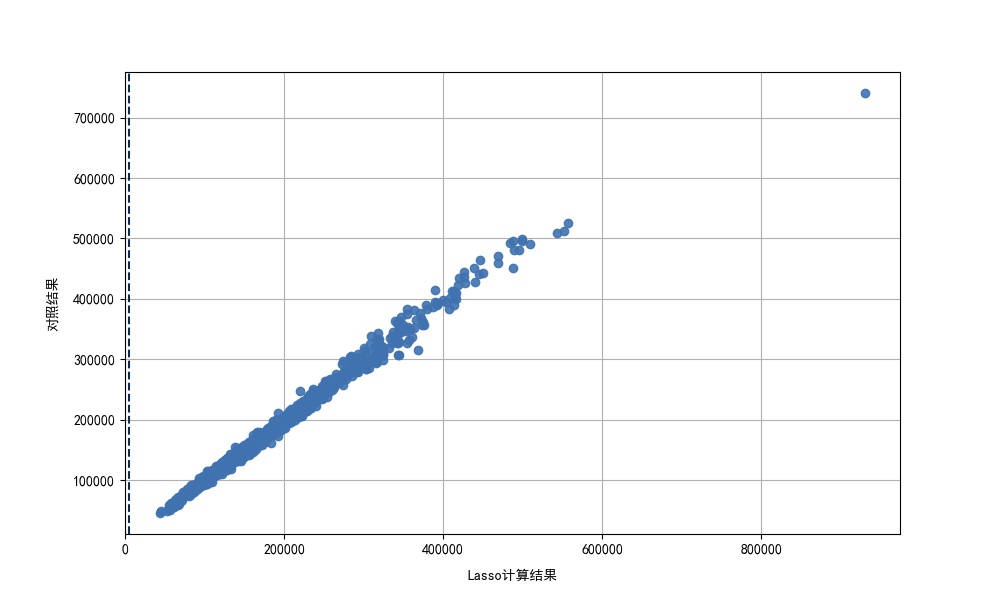


2、采用Lasso方法

方法介绍：

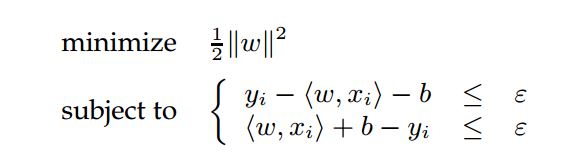


结果表现：

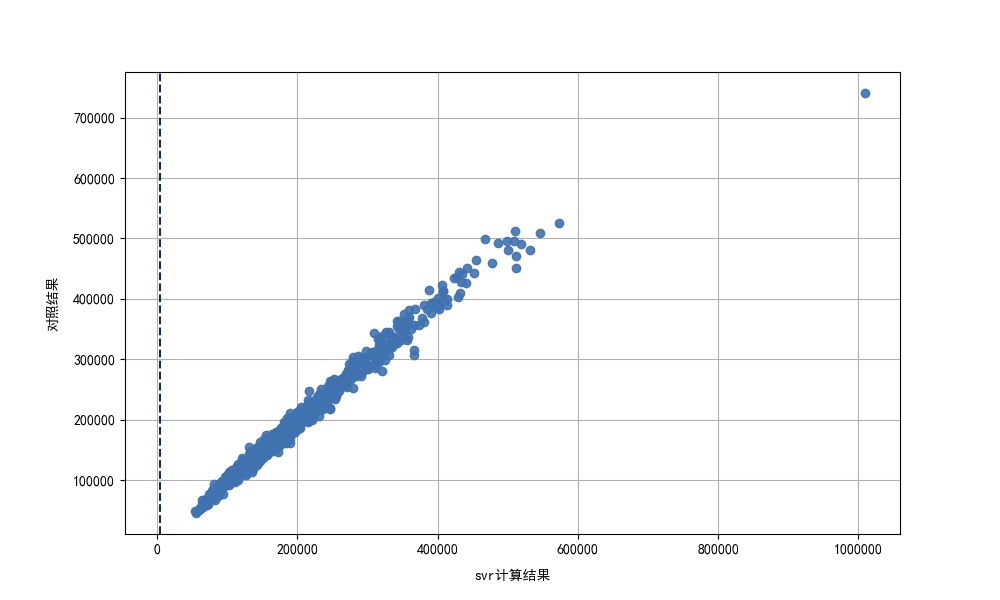


3、SVR方法

方法介绍：



表现：



4、*LGBM方法*

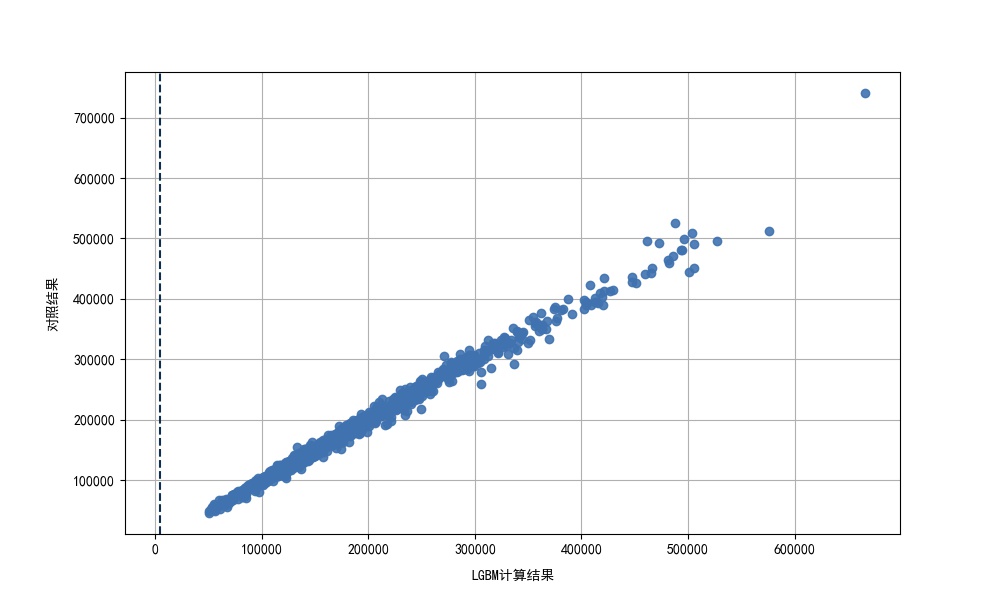
*方法介绍：*

*定义L为损失函数，有*

*(x)*

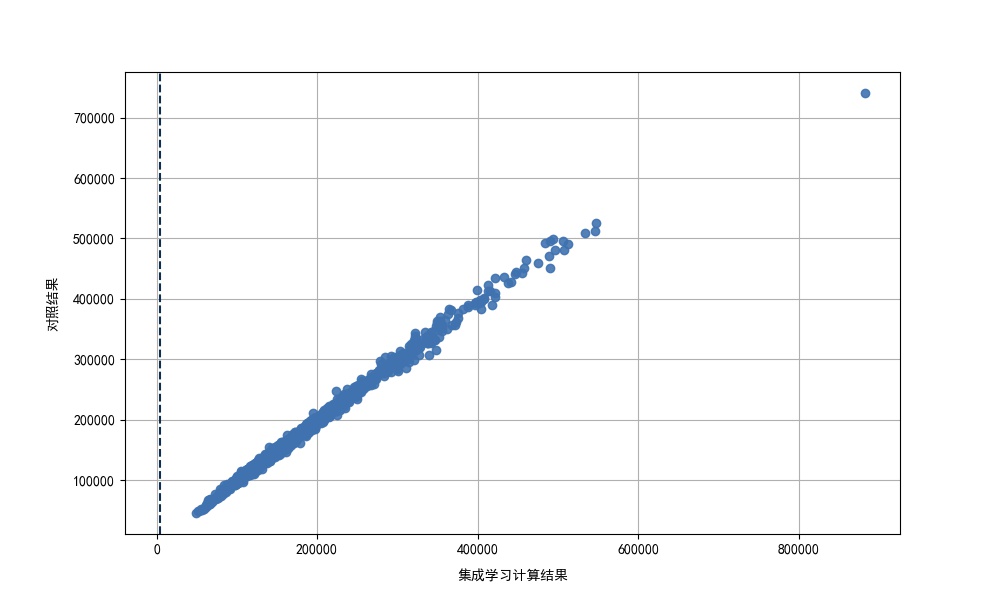
*求*的值

*表现：*



对上述方法结果做集成学习，在所得系数上做加权平均。一次来抵消不同模型所带来得偏差。

结果表现：



通过数字可视化的结果展示，我们可以发现集成学习能够带来更优更稳定的预测结果。由于。

第四部分 总结

通过上述探索，我们发现机器学习从某种意义来说其实是一种工程，单靠人为的调参很难有稳定的结果。而工程应用中，比较稳定的准确率比某种情况下的高准确率更为重要。要做到这一点，首先要建立工程化的机器学习框架与多模型并用方式。在上述方法中，我们从场景——数据——数据处理——特征工程——模型分析与选择——交叉验证——集成学习的方式极大地方便了我们将机器学习的应用于现实工程应用中。

