IAI-classification

TODO-2,3: Train, Tune and Evaluate Model

观察到数据的正负例非常不均匀，我先写了一个"全阴性"分类器(AN)，对所有样本都判断为阴性，AN分类器在训练集3781个样本上的正确率有90.55，也就是负例（阴性）和正例（阳性）的比例竟然达到了9.58的比例，这对于很多分类算法是非常不利的，因为很多分类算法都是基于各个类别出现的概率均匀分布。

我先测试了几种基础的分类算法, Logistics Regression(LG)，Decision Tree (DT)，Kernel SVM（KSVM), Random Forest(RF), improved K-Nearest-Neighbors(MyKNN,改良knn)

其中KNN是我自己实现的，先对样本点进行去除重复点操作，并采用kd-tree存储样本点集来降低查询复杂度，计算概率时使用距离反比加权，并对样本种类进行加权，以减小正负例的差异，具体计算点v_i为正例的概率采用的公式是：$p(y_i=1)=\frac{\sum_{X_j,y_j\in S,y_j=1}wy_j/dist(X_i,X_j)}{\sum_{\forall X,y\in S}(1+(w-1)I[y=1])y/dist(X_i,X)}$

其中Y为训练集合的label，dist为两点的平方距离。

经过简单测试，我发现这些模型的正确率都 不如AN分类器，特别是Gussian Naive Bayes分类器基本完全失效。我意识到单纯用正确率评价模型是无意义的，因为在样本中负例对正例的影响太大了。结合TODO-3的提示，我写了一个评估模型函数，它主要评估模型的指标有**Accuracy, Precision, Recall, Confussion Matrix, F1-Score，ROC曲线和AUC值**，对每个模型，相对于正确率，我们更看重他的recall，因为我们比较关注真阳性样本中有多少样本能被预测出来。

为了得到比较好的参数，我采用交叉验证的方式进行调参，调整的参数主要是class_weight。选取的指标是交叉验证平均正确率大于50%时的recall——我们调整分类器的参数，让它们的recall达到最大值且正确率不低于50%。

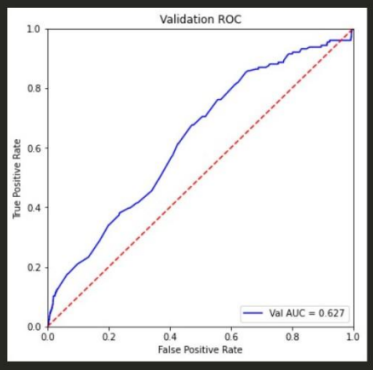
实验对上述模型进行调参，发现这些模型f1-score达到极大值的时候，正确率会出奇的低，原因在于如果要多考虑阳性样本，阴性样本的正确率就会降低，导致总正确率非常低，为了能评估模型，我界定了一个范围，就是验证正确率不低于70%的条件下进行选取。

对LR，DT， KSVM，RF，MyKNN 五种模型通过测试集合进行评估。

测试结果如下：

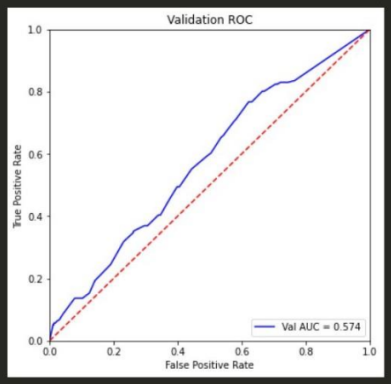
Logistic Regression

```
evaluation of Logistic Regression:
Accuracy of Logistic Regression: 0.5775630703166935
Confussion Matrix of Logistic Regression:
[[968 719]
 [ 68 108]]
Precision score of Logistic Regression: 0.13059250302297462
Recall score of Logistic Regression: 0.6136363636363636
F1 score of Logistic Regression: 0.2153539381854437
```



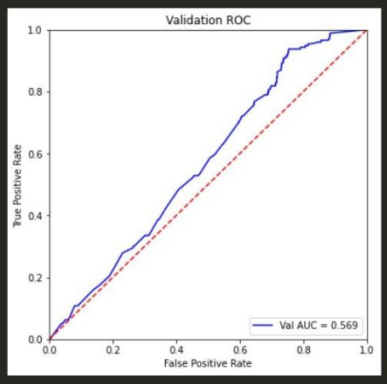
Decision Tree

```
evaluation of Decision Tree:
Accuracy of Decision Tree: 0.512614063338701
Confussion Matrix of Decision Tree:
[[850 837]
 [ 71 105]]
Precision score of Decision Tree: 0.11146496815286625
Recall score of Decision Tree: 0.5965909090909091
F1 score of Decision Tree: 0.1878354203935594
```



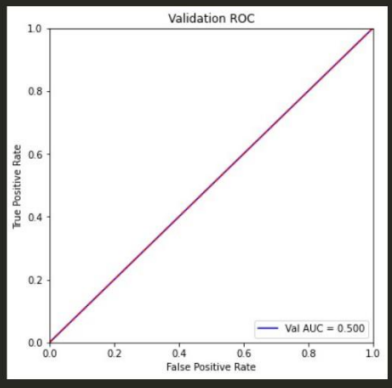
Random Forest

```
evaluation of Random Forest:
Accuracy of Random Forest: 0.49114331723027377
Confussion Matrix of Random Forest:
[[810 877]
 [ 71 105]]
Precision score of Random Forest: 0.10692464358452139
Recall score of Random Forest: 0.5965909090909091
F1 score of Random Forest: 0.18134715025906736
```



All Negative

```
evaluation of All Negative:
Accuracy of All Negative: 0.90552871712292
Confussion Matrix of All Negative:
[[1687    0]
 [ 176    0]]
Precision score of All Negative: 0.0
Recall score of All Negative: 0.0
F1 score of All Negative: 0.0
```

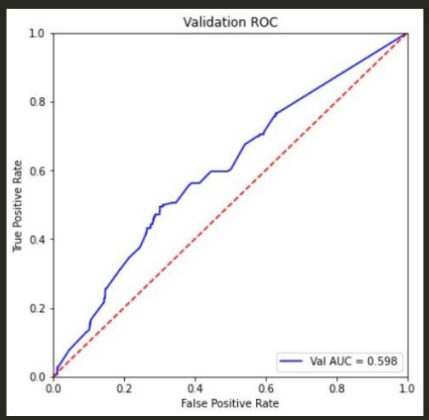


Support Vector Machine

```
evaluation of Support Vector Machine:
Accuracy of Support Vector Machine: 0.90552871712292
Confussion Matrix of Support Vector Machine:
[[1686    1]
 [ 175    1]]
Precision score of Support Vector Machine: 0.5
Recall score of Support Vector Machine: 0.0056818181818
F1 score of Support Vector Machine: 0.01123595505617977
```

Improved K-Nearest-Neighbors

```
evaluation of Improved K Nearest Neighbors:
Accuracy of Improved K Nearest Neighbors: 0.6714975845410628
Confussion Matrix of Improved K Nearest Neighbors:
[[1164  523]
 [  89   87]]
Precision score of Improved K Nearest Neighbors: 0.1426229508
Recall score of Improved K Nearest Neighbors: 0.4943181818181
F1 score of Improved K Nearest Neighbors: 0.22137404580152673
```



从上述结果可以看出效果比较好的模型有逻辑回归和KNN等，前者在保证正确率的情况下有较高的Recall，后者在保证Recall的情况下有较高的正确率。

尝试：过采样和欠采样

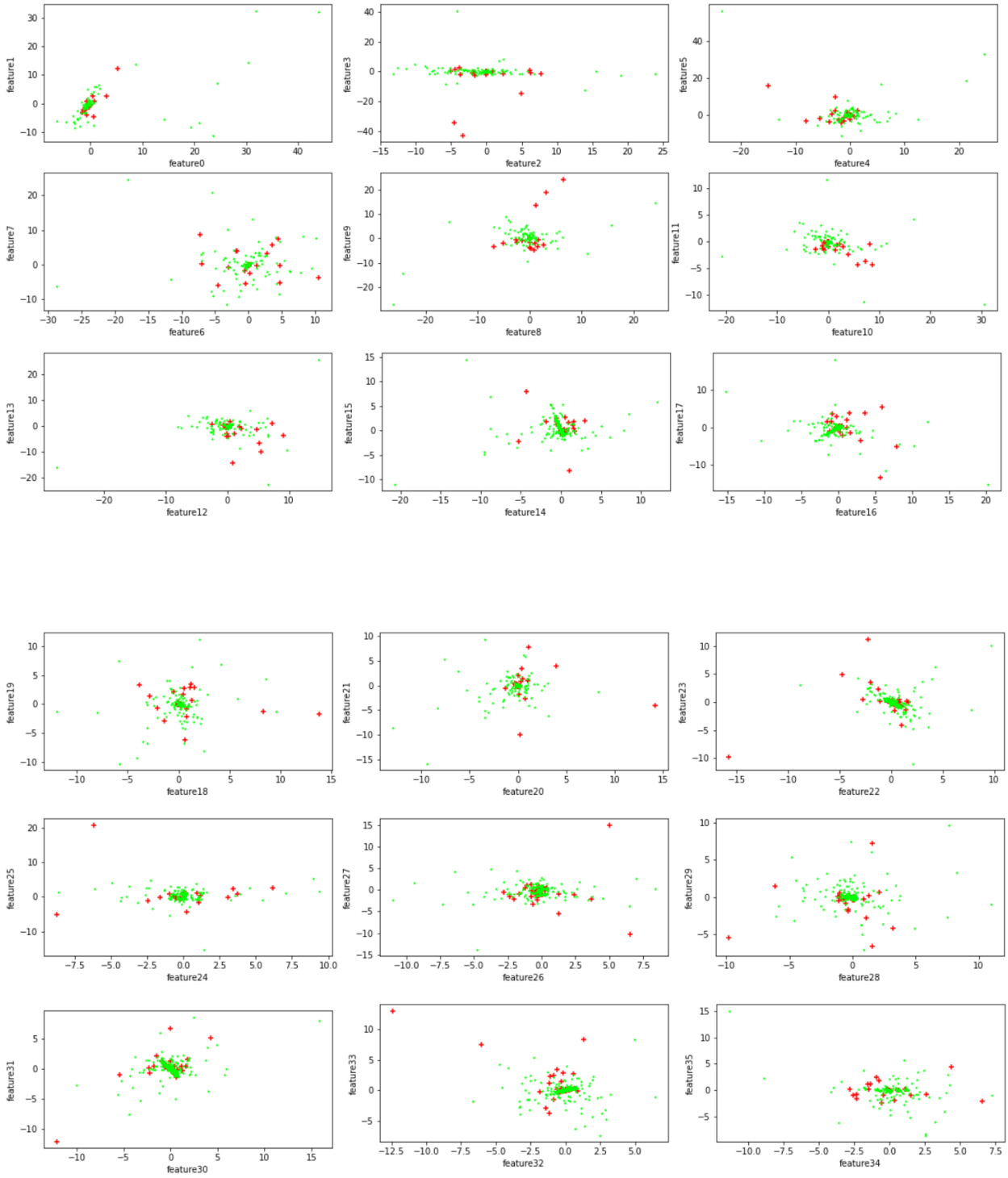
为了解决正负例不均匀的问题，我还采用了过采样和欠采样对样本进行处理。

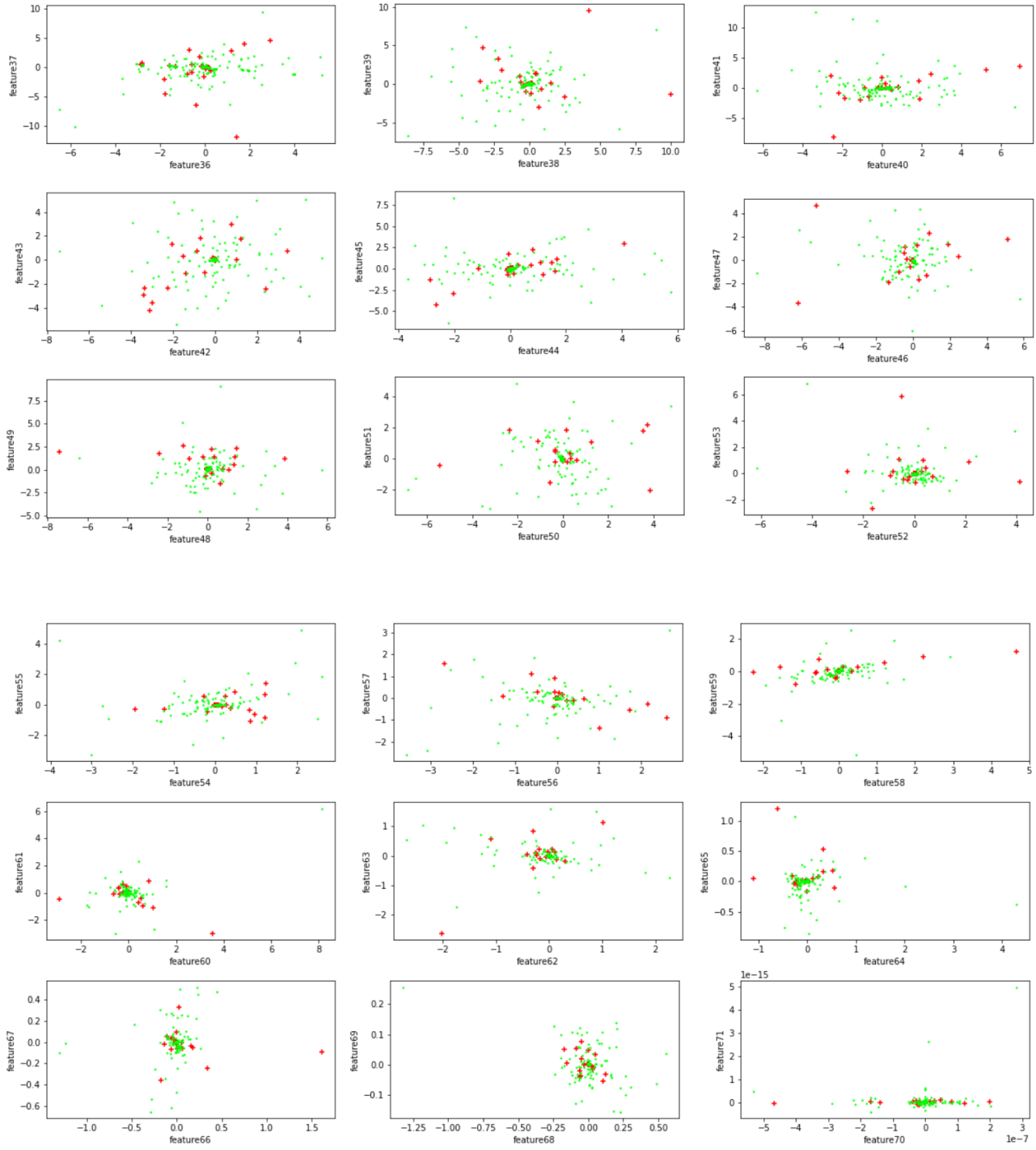
过采样：在训练集，我将原有的正例随机复制扩充达到和负例一样的规模，使得正负例在训练集中较均匀分布。但实际训练的效果并不好。

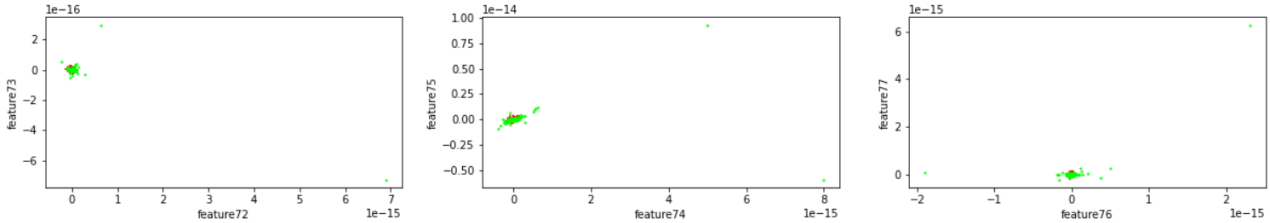
欠采样：我尝试将负例随机取样到和正例一样的规模，但由于训练集缺失过多，训练效果反而更差了。非常令人绝望。

思考：模型的可分性

可以看出，最终几种模型的评估结果都不是很好，我尝试了很多种改进方式，都并不能达到可观的结果。于是，绝望的我决定从特征集入手，我决定把特征处理过的点用多图画出来。




feature37
feature36
feature39
feature38
feature41
feature40
feature43
feature42
feature45
feature44
feature47
feature46
feature49
feature48
feature51
feature50
feature53
feature52
feature55
feature54
feature57
feature56
feature59
feature58
feature61
feature60
feature63
feature62
feature65
feature64
feature67
feature66
feature69
feature68
feature71
1e−15
feature70
1e−7



该图绿色的点表示的负例子，红色的'+'形点代表的是正例，这里为了让正例更加明显，我调大了正例点的大小。可以看出，在各个特征维度，正负例的中心点都是接近的，很难将它们较好的分离，我们模型能够保证较好的Recall就比较不错了。

TODO-1 Feature Engineering

对原始特征的处理我主要采用了PCA技术，在学习基本原理后我用numpy实现了PCA的完整过程，并且与sklearn的库进行比较，证明了自己实现过程的正确性。

本部分我主要采用了四种特征选取：

- 原始数据`dataset_raw`
- 删除强相关性的列`dataset_rm`
- 删除强相关性的列后标准化`dataset_rm_std`
- 删除强相关性的列后进行PCA `dataset_rm_pca`，取映射维度为60

我们使用逻辑回归和决策树对四中数据集进行评估, 评价指标为正确率和Recall(超参数采样第二、三部分选取出的参数)。

(Acc, Recall)	`dataset_raw`	`dataset_rm`	`dataset_rm_std`	`dataset_rm_pca`
LR	(0.567,0.653)	(0.572,0.642)	(0.578,0.614)	(0.578,0.614)
DT	(0.526,0.619)	(0.509,0.619)	(0.514,0.618)	(0.510,0.597)

可以看出删除强相关数据后数据正确率有所提升，但采用PCA后各个模型并无明显的变化，说明数据的特征维度并不适合用PCA进行降维。

TODO-4: Predict other labels

此部分完成在`predict_task.ipynb`，我选取的label为`patient_addmited_to_regular_ward_(1=yes,_0=no)`。

虽然在阴阳性预测上我们遭遇了失败，但这个过程帮助我们了解了数据挖掘和分类的全流程。

在经过数据预处理（填充缺失值），我采用的特征工程是消除相关列。

模型选取上我选择的是逻辑回归和决策树两个分类器。评估的指标有正确率，precision,recall,f1-score等等。

使用交叉验证进行调参，使得模型正确率在训练集上达到最高，LR和DT树的正确率都达到了98%，但是一旦把测试数据放进最佳模型，正确率却降下来了，以下是我的训练结果：

	LR	**DT**
训练时正确率	0.984	0.981
测试时正确率	0.949	0.936

将调参的指标换成f1-score后得到类似的结果（训练和测试差距较大），经过分析和实验，我找出了训练结果和验证结果差距较大的原因：因为本部分我们没有随机取样，**正负例在训练集和测试集的分布并不均**，经过计算：测试集合的正例比例约为1%，而测试集中正例的比例高达4.51%，如此差距导致训练的模型很难在测试集中有较好的表现，想要提高正确率只能改变取样方式。