机器学习纳米学位

毕业项目

Hanwei Zhu

2018年4月20日

I. 问题的定义

项目概述

随着人工智能的发展，人机语言交互已经成为重要的研究领域之一。Apple公司的Siri，Microsoft的Cortana，Google的Google Assistant和Amazon的Alexa等等开发的虚拟AI助手都已经投入商用并取得了巨大的成功。这都是以语音识别技术作为基石。可以想象未来基于语音识别的各种场景：对话的方式来完全地取代人工键盘输入；声控智能家居；声纹加密；在现有的技术支持下实现这些场景似乎已经指日可待。

现有的研究指出传统的语音交互具有距离近、抗干扰能力弱、不可持续等特点。随着算力的提升和人工智能研究的发展，现有的机器学习，模式识别技术等等都能够很好地改进上述缺点。这也是本次项目探索的重点和方向。

本次毕业项目将实现一个简单的语音识别器：对不同的声音样本进行性别识别。该项目基于Kaggle竞赛赛题“Gender Recognition by Voice”。根据Kaggle上竞赛举办方提供的声音数据集来搭建识别系统，能够准确地识别采集到的声音属于男性还是女性。

问题陈述

针对语音识别问题，本次项目将采用现有的机器学习算法——尤其是监督学习来解决。对于让计算机学习如何正确辨别男性还是女性，该问题本质上属于监督学习中基本的二元分类问题：即根据输入的一系列特征输出二元标签。在给出人工标签的数据集上训练模型，通过减小和标签差距的方式来找到模型的最佳参数。

本次项目主要参考了[1]，并结合在纳米学位的课程中学习到的标准数据处理流程进行实验。具体实验设计如下：

- 数据预处理

  包括数据清洗，标签编码，数据标准化，将数据集拆分成训练集，验证集和测试集。

- 模型训练

  可选择的模型包括：逻辑回归，决策树，随机森林，支持向量机，神经网络，XGBoost等等。并使用交叉验证的方式进行参数选择。

- 模型评估

  使用f1-score，混淆矩阵等对模型的预测结果进行评估。

- 结果可视化

  使用可视化工具对结果进行可视化分析。

最后我将通过使用预先准备好的测试数据来衡量每一个模型的指标。现有的机器学习算法和在Kaggle上参赛者上传的示例均说明了在现有的技术条件下这一问题能够得到很好地解决。我期望的结果能够在测试集上达到识别准确率超过95%准确率这一评价指标。

评价指标

根据观察，现有数据集的特点是男女两类标签的样本数目一致。也就是说准确率accuracy能够很好地当做测试指标。计算方法为：

准确率 = 模型判断正确的样本数量 / 样本总数

对于均匀分布的数据集，准确率已经足够说明模型的好坏。本次项目将完全根据准确率来对分类模型进行评估。

II. 分析

数据的探索

本次项目使用了Kaggle竞赛“Gender Recognition by Voice”提供的数据集进行实验。现有的数据集提供了`voice.csv`文件。`voice.csv`文件内一共包含了3168个样本，其中50%为男性，50%为女性。其中每条记录包括以下信息：

- meanfreq: 频率平均值 (in kHz)
- sd: 频率标准差
- median: 频率中位数 (in kHz)
- Q25: 频率第一四分位数 (in kHz)
- Q75: 频率第三四分位数 (in kHz)

- IQR: 频率四分位数间距 (in kHz)
- skew: 频谱偏度
- kurt: 频谱峰度
- sp.ent: 频谱熵
- sfm: 频谱平坦度
- mode: 频率众数
- centroid: 频谱质心
- peakf: 峰值频率
- meanfun: 平均基音频率
- minfun: 最小基音频率
- maxfun: 最大基音频率
- meandom: 平均主频
- mindom: 最小主频
- maxdom: 最大主频
- dfrange: 主频范围
- modindx: 累积相邻两帧绝对基频频差除以频率范围
- label: 男性或者女性

以上特征都是一段语音进过筛选后的重要信息，已经通过了R语言脚本对原始的音频进行了预处理[2]。直接从csv文件读取数据，数据集基本信息如下所示。可以看出提供的数据集比较干净，没有出现脏数据/缺失数据的情况。但是可以看出数据区间各不相同，我们需要对其进行归一化处理。

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3168 entries, 0 to 3167
Data columns (total 21 columns):
meanfreq    3168 non-null float64
sd          3168 non-null float64
median      3168 non-null float64
Q25         3168 non-null float64
Q75         3168 non-null float64
IQR         3168 non-null float64
skew        3168 non-null float64
kurt        3168 non-null float64
sp.ent      3168 non-null float64
sfm         3168 non-null float64
mode        3168 non-null float64
centroid    3168 non-null float64
meanfun     3168 non-null float64
minfun      3168 non-null float64
maxfun      3168 non-null float64
meandom     3168 non-null float64
mindom      3168 non-null float64
maxdom      3168 non-null float64
dfrange     3168 non-null float64
modindx     3168 non-null float64
label       3168 non-null object
dtypes: float64(20), object(1)
memory usage: 519.8+ KB
```

通过Pandas基本统计量分析，得到的统计量如下：

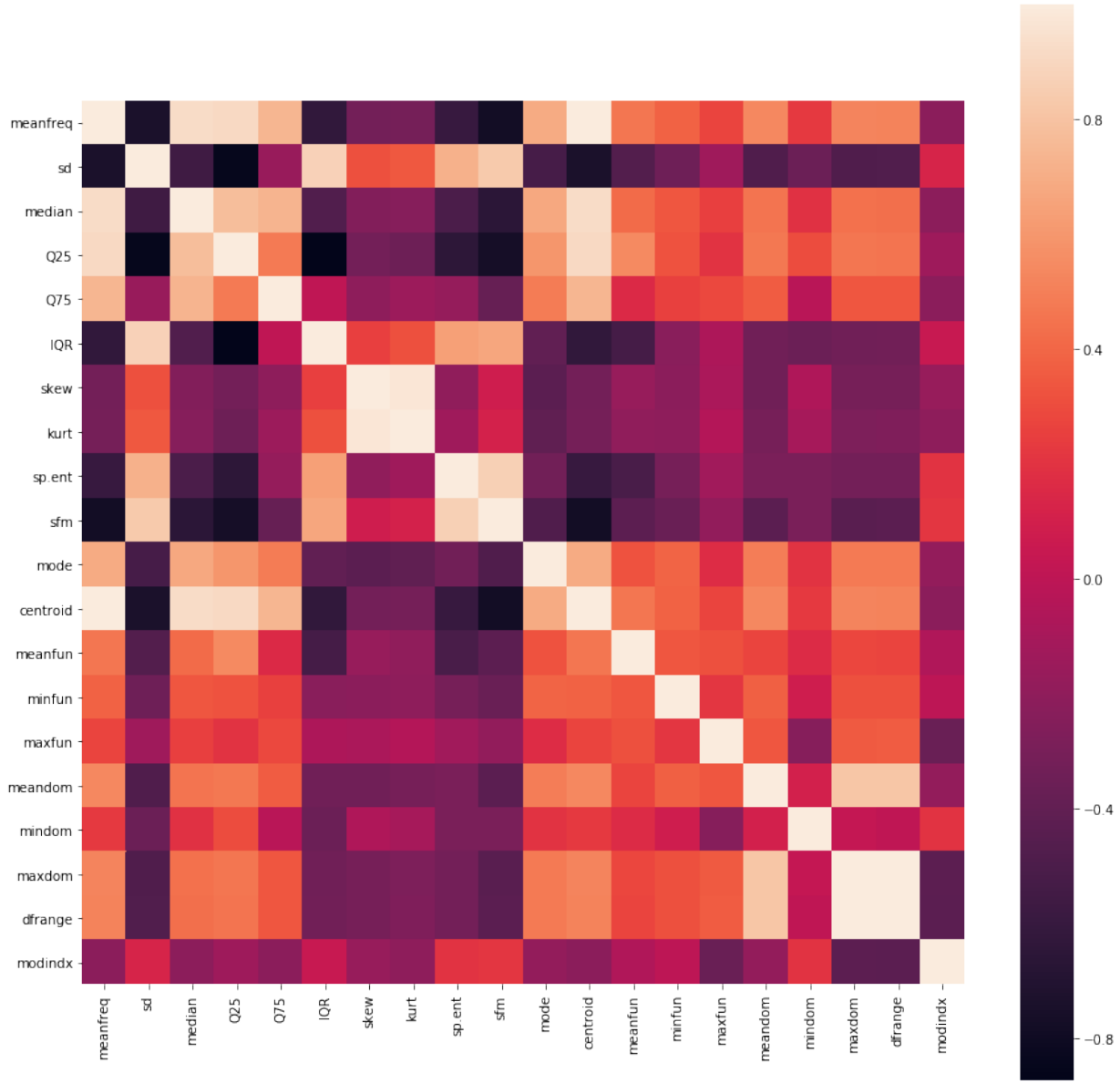
	count	mean	std	min	25%	50%	75%	max
meanfreq	3168.0	0.180907	0.029918	0.039363	0.163662	0.184838	0.199146	0.251124
sd	3168.0	0.057126	0.016652	0.018363	0.041954	0.059155	0.067020	0.115273
median	3168.0	0.185621	0.036360	0.010975	0.169593	0.190032	0.210618	0.261224
Q25	3168.0	0.140456	0.048680	0.000229	0.111087	0.140286	0.175939	0.247347
Q75	3168.0	0.224765	0.023639	0.042946	0.208747	0.225684	0.243660	0.273469
IQR	3168.0	0.084309	0.042783	0.014558	0.042560	0.094280	0.114175	0.252225
skew	3168.0	3.140168	4.240529	0.141735	1.649569	2.197101	2.931694	34.725453
kurt	3168.0	36.568461	134.928661	2.068455	5.669547	8.318463	13.648905	1309.612887
sp.ent	3168.0	0.895127	0.044980	0.738651	0.861811	0.901767	0.928713	0.981997
sfm	3168.0	0.408216	0.177521	0.036876	0.258041	0.396335	0.533676	0.842936
mode	3168.0	0.165282	0.077203	0.000000	0.118016	0.186599	0.221104	0.280000
centroid	3168.0	0.180907	0.029918	0.039363	0.163662	0.184838	0.199146	0.251124
meanfun	3168.0	0.142807	0.032304	0.055565	0.116998	0.140519	0.169581	0.237636
minfun	3168.0	0.036802	0.019220	0.009775	0.018223	0.046110	0.047904	0.204082
maxfun	3168.0	0.258842	0.030077	0.103093	0.253968	0.271186	0.277457	0.279114
meandom	3168.0	0.829211	0.525205	0.007812	0.419828	0.765795	1.177166	2.957682
mindom	3168.0	0.052647	0.063299	0.004883	0.007812	0.023438	0.070312	0.458984
maxdom	3168.0	5.047277	3.521157	0.007812	2.070312	4.992188	7.007812	21.867188
dfrange	3168.0	4.994630	3.520039	0.000000	2.044922	4.945312	6.992188	21.843750
modindx	3168.0	0.173752	0.119454	0.000000	0.099766	0.139357	0.209183	0.932374

探索性可视化

- 相关系数可视化

数据集中各项特征的*相关系数*经过可视化后的热力图如下，颜色深浅代表了相关系数的高低：颜色越浅表示正相关程度越高；颜色越深表示负相关程度越高。

可以看出特征*meanfreq*和 *median*，*meanfreq*和*Q25*，*meanfreq*和*centroid*，*median*和*centroid*，*Q25*和*centroid*，*dfrange*和*maxdom*，*skew*和*kurt*这几组特征呈现出较高的相关性（±0.9左右）。



- 将数据Normalize后进行PCA降维可视化。

x轴，y轴代表了降维后的坐标。方块的点代表了女性声音样本；圆点表示男性声音的样本。可以看出降成两维后提供的信息还不能够对两者明显地区分：图像左上方位比较多男性样本分布；右侧女性样本较多。但是很多样本大多还处于两者重叠的位置。

算法和技术

逻辑回归

逻辑回归算法是一种在机器学习中常见的分类算法。其基本思想是将特征进行线性变换后，输入sigmoid函数进行非线性变换，将结果压缩至[0, 1]区间内。从而达到分类的目的。使用逻辑回归的好处是模型简单却有可能取得很好的训练效果。

我们可以赋予声音信息的每个特征一个权重并将结果相加后，用sigmoid函数进行非线性变换。再以交叉熵作为损失函数对预测标签（模型预测是男性或女性）和真实标签（实际是男性或女性）作为损失函数，最后用梯度下降的方式不断调节各项权重直至损失函数收敛。

sklearn里直接提供了sklearn.linear_model.LogisticRegression模型，能够自动生成逻辑回归模型。照耀可供调节的参数有：

Penalty：规则化范数，可以选择“L1”范数（Lasso regularization）和“L2”范数（Ridge Regression）

C：正则化权重，默认为1

随机森林

单个决策树是一种简单，直观又高效的分类方法。通过不断对数据集中的样本进行拆分，最后得到分类结果。但是决策树带来的问题是非常容易对于训练集过拟合，可以通过剪枝等手段来防止过拟合。随机森林[2]是一种高效的ensemble learning方法。具体算法是通过bagging的方式将多个决策树集合在一起进行训练和决策。这样既能将单个决策树的预测能力集合起来，又能有效地防止训练单个决策树过拟合的问题。

与随机树不同，我们先对输入数据进行行采样和列采样。随后在分裂节点的时候，我们不断通过输入样本的特征进行询问（例如：meanfreq是否小于0.3？sd是否小于0.9？），最后得出是否是男性还是女性的结论。最后在预测阶段，我们集合起所有的决策树进行投票表决，通过多数投票来决定该样本的标签。

sklearn里也提供了sklearn.ensemble.RandomForestClassifier模型，能够自动生成逻辑回归模型。主要可供调节的参数有：

max_features：单个决策树使用的最大特征数

n_estimators：决策树的数量

max_depth：单个决策树的最大深度

XGBoost

同样是对单个决策树进行emsemble，但是不同于随机森林的bagging的方式：XGBoost是gradient boosting decision tree的一种实现[3]。通过梯度下降，正则化的方式来训练模型和控制模型复杂度。XGBoost作为一种高效的集成学习方法出现在各大数据竞赛和实际应用中，其既可以用于分类也可以用于回归问题中。

主要可供调节的参数有：

max_depth：单个决策树的最大深度

n_estimators：决策树的数量

learning_rate：梯度下降的学习率

基准模型

实验采用naive分类器——总是预测输入为男性（或女性）作为基准模型。这意味着naive分类器的accuracy能够达到50%（因为数据集中男性和女性的样本各占了50%）。如果我们的模型能够大于这一评估指标，说明模型至少比随机预测（50%）的效果要好。

III. 方法

数据预处理

如上所述，得到基本统计量后对数据进行预处理：

- 使用sklearn的normalize包对实验数据标准化
- 将数据集拆分为训练集（80%）和测试集（20%），同时将随机种子设定为0确保每次实验得到相同的结果。

因为根据数据探索的结果，没有出现数据缺失，极端异常值的情况。所以除了标准化以外不需要进行额外的处理。

执行过程

- 模型构建

直接使用sklearn构建了逻辑回归，随机森林模型

使用XGBoost官网的安装包在试验环境（Ubuntu）中进行安装编译成Python package，成功在Python环境里构建了XGBoost模型。（具体安装过程见Readme）

最后试验中将构建函数使用K-fold（K=10）对模型进行交叉验证：将训练集拆分成9个测试子集和1个验证子集，交叉验证模型的正确性。通过K-fold分别对各个模型进行参数选择，代码如下：

```
from sklearn.model_selection import KFold
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer


def fit_model(X, y, clf, params):
    cross_validator = KFold(n_splits=10)
    scoring_fnc =
make_scorer(metrics.accuracy_score)
    grid = GridSearchCV(clf, params,
scoring=scoring_fnc, cv = cross_validator)
    grid = grid.fit(X, y)
    return grid.best_estimator_
```

完善

使用了网格搜索技术，最终得到的最优模型的参数如下:

- Logistic regression

调整参数为：

penalty: l1, l2;

C: 0.1, 0.3, 0.5, 0.7, 0.9

经过试验后最终最优模型参数如下：

```
LogisticRegression(C=0.3, class_weight=None,
dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100,
multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None,
solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

- Random forest

调整参数为：

criterion: 'gini', 'entropy';

n_estimators: 5, 10, 15, 20;

max_depth: 1~10

经过试验后最终最优模型参数如下：

```
RandomForestClassifier(bootstrap=True,
class_weight=None, criterion='entropy',
            max_depth=9, max_features='auto',
max_leaf_nodes=None,
            min_impurity_decrease=0.0,
min_impurity_split=None,
            min_samples_leaf=1,
min_samples_split=2,
            min_weight_fraction_leaf=0.0,
n_estimators=15, n_jobs=1,
            oob_score=False, random_state=None,
verbose=0,
            warm_start=False)
```

- XGBoost

调整参数为：

max_depth: 2~10; **n_estimators**: 10 ~ 130; **learning_rate**: 0.01 ~ 0.4

```
XGBClassifier(base_score=0.5, booster='gbtree',
colsample_bylevel=1,
       colsample_bytree=1, gamma=0,
learning_rate=0.3, max_delta_step=0,
       max_depth=2, min_child_weight=1,
missing=None, n_estimators=130,
       n_jobs=1, nthread=None,
objective='binary:logistic', random_state=0,
       reg_alpha=0, reg_lambda=1,
scale_pos_weight=1, seed=None,
       silent=True, subsample=1)
```

IV. 结果

模型的评价与验证

经过上述参数调整后，使用测试集对实验结果进行最终测试。观察比较各个模型的最终指标：

	Logistic Regression	Random Forest Classifier	XGBoost
Accuracy	0.9700315457413249	0.9763406940063092	0.9826498422712934

比较看来，XGBoost效果是最好的，也和之前的预期相符。但是值得注意的是简单常见的单个逻辑回归模型也能够达到97%的准确率，已经非常接近另外两个ensemble的复杂模型。说明了模型未必一定是越复杂越优异，还要考虑到训练时间成本，硬件条件等等（逻辑回归模型训练时间远远短于其他两者）。

因为训练集和测试集使用了sklearn中的train_test_split方法，所以拆分的结果可能并不一致。每次训练集的不同可能会在Accuray上的表现结果有略微的差异。可以通过设置随机种子的方式来固定实验结果（本次实验设置随机种子为0）。

合理性分析

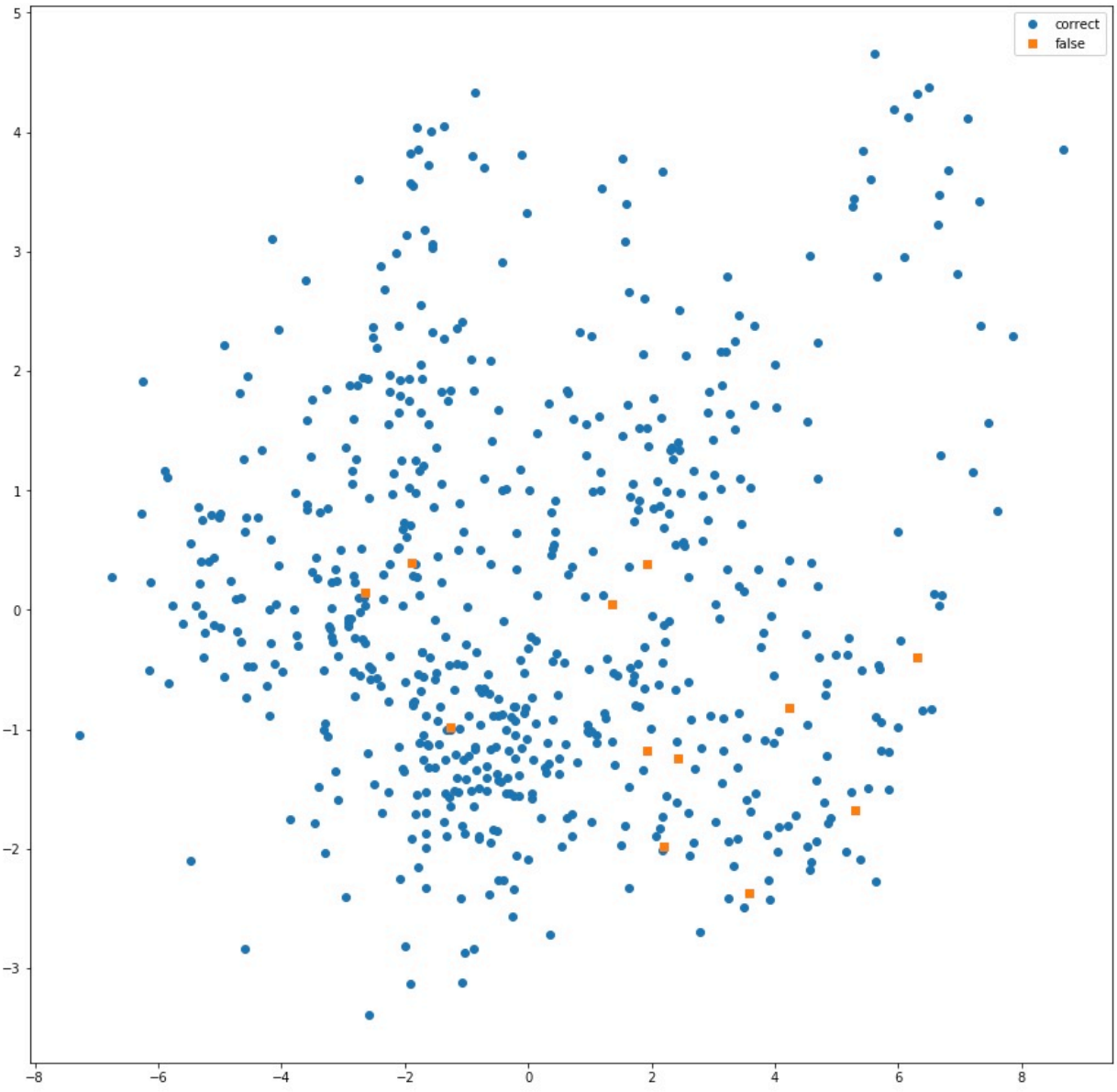
同我们先前设置的基准进行比较，我们训练的三个模型都远远超过了naive基准模型的指标（50%）。这一结果说明了我们构建的模型都比随机预测（50%）的效果远远要好，也证明了我们训练的结果是十分可靠的。本次实验和先前的期待的结果完全一致，甚至在测试集上的表现来看高于预期。

V. 项目结论

结果可视化

以下讨论XGBoost最优模型的预测结果可视化。

将测试集经过降维后Plot在2D平面上。圆点表示我们的模型分类正确的样本，方形表示分类错误的样本。可以看出分类错误的点比较均匀分散，没有呈现特定的趋势。



对项目的思考

本次项目中我熟悉了整个数据挖掘的基本流程。从数据预处理，到模型选择，构建，训练和评估，可视化等等都在本次项目中有所体现。

通过查阅各种资料，我很大程度上提高了在机器学习模型构建上的动手能力。查阅资料能够学习到很多有意思的方法和思路，遇到和学习过的方法相似的地方也能够重新巩固自己的知识。

困难而枯燥的的是参数选择和训练部分。网格搜索运行时间长，并且随着参数的增加呈指数增长；还有由于随机森林，XGBoost等模型属于ensemble learning的方法，其训练也十分耗时。

从实验结果来看，本次项目构建的分类器是十分成功的。即使投入实际应用，这样一个简单的男女性识别也能够达到很高的准确率。

需要作出的改进

本次项目只是粗略地构建了一个分类器，尚有很多需要改进的地方。例如如何能够投入真正的应用中去：现在的分类器只是一个简单的离线应用，如何能够改进成在线应用，甚至改造成online learning的学习模型都是值得探究的问题（是否会影响模型收敛，训练时间如何等等）。还有如何在服务器上部署，训练机器学习模型也是一个非常值得探讨的话题。由于本次项目数据量很小，所以仅仅在单台节点上就能很快地完成训练和预测。如果在一些大型的应用场景下数据吞吐量是非常大的，我们可能需要部署集群进行训练。这些方案都可以改进项目的分类器，但是同时也引入了更多问题让我们进行思考。

- ## Reference

[1] Nasrabadi, N. M. (2007). Pattern recognition and machine learning. *Journal of electronic imaging*, *16*(4), 049901.

[2] Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.

[3] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794). ACM.