accuracy1.py:切割音频，通过对音频信息的提取对比标签，获取判断错误列表

beat.py:根据音频能量的突变来对节奏拍子进行提取

beat\_cut.py: 根据音频能量的突变来对节奏拍子进行提取,并根据节奏剪切音频，将一连串的音频剪切为多个单音

cal.py:将钢琴基声调数据转化为音符数据

change\_amplitude.py:改变音频格式与波特率

change\_form.py:转换音频格式

chord\_create.py:用时域相加的方法将两个以上的单音组合起来，生成新的和弦，为我们后续和弦训练做铺垫

correlation.py:这是我们测试观察使用最多的文件，我们用它来观察音频文件的时域波形以及傅里叶变换后的频域图像以及在各频域峰值的峰值检测的图像，用来我们人为地分析音频的特征

cut.py:剪切音频到合适的长度

denoise.py:原理和correlation.py类似，在这里加入了频域理想滤波，滤去了高频噪声分量

fail\_correlation.py:用来分析错误的样本的时域与频域波形，并根据合适的滤波方法生成新的音频文件

frequence.py: 读取完整的帧数据到str\_data中,将波形数据转换为数组，修改采样点数，修改采样点数和起始位置进行不同位置和长度的音频波形分析

jiqixuexi.py:用来测试本地机器学习库的安装完成情况

peak\_out.py:用来生成.json文件，文件中包含了我们所有的根据数据集音频生成的数据集时域与频域的样本数据，便于后续机器学习单音识别、错音识别、和弦识别等模型的训练

pitch.py:用wave库中的函数直接得出音频的频率信息，准确率较低

preprocure.py:测试使用时域自相关函数的方法得出音频文件的频率

rate\_cul.py:用来测试我们的频率计算方法的正确率

read.py:读取文件

record.py:根据我们需要的格式与波特率自行录制数据样本集

single\_sound\_denoise.py:对单音进行高频滤波滤除噪声

timedata\_out.py:输出并保存音频文件的时域数据

transform.py: 将钢琴基声调数据转化为音符数据

videowave.py:将幅值归一化后输出保存音频文件

backend.py:本地搭建后端服务器，用以处理小程序之间的交互，根据小程序的POST和GET请求运行不同的函数，处理小程序的数据与文件信息，并将处理后的结果返回给小程序，包括单音识别、错误类型识别、乐曲识别、AI谱曲等

小程序端：wxml ui界面与js文件控制功能逻辑，上传文件到后端服务器并从后端服务器获取返回的结果

aixiege.py：从网上下载乐器库后调用乐器库并用事先准备好的音符、节奏数组或者ai谱曲后生成的数组来生成音乐并保存

.ipynb文件: 运行于jupyternotebook的识别神经网络模型