FlyAI算法竞赛¥20,000邀您加入！

FlyAI|积分+现金，竞赛赚年薪！

> 深度学习，赚钱有趣！
>
> FlyAI是一个为算法工程师提供（深度学习）项目竞赛并支持GPU离线训练的网站。我们每周提供不同领域的项目竞赛并附带代码样例供您参考学习。样例所使用开发框架涉及TensorFlow、Keras、PyTorch.
>
> 通向百万年薪的挑战之路，在FlyAI与优秀算法工程师一起学习、探讨前沿技术！

> 四大特点

- 高质量的数据集、多领域的开源项目案例
  - 项目涉及领域：自然语言处理、图像识别、语音识别等
  - 每周更新高质量项目专属代码样例,免费下载查看
  - 支持多平台运行，一键配置学习环境
- 多卡GPU资源 一键使用
  - 提供强大算力，快速迭代模型质量
  - 一键提交离线训练服务，及时通知模型训练进度
- 能力变现、竞赛式项目实力打榜
  - 挑战项目刷新排行榜，赢得高额悬赏
  - 使用不同深度学习框架验证，提升自己的算法能力
- 全行业的数据化及AI需求发布平台
  - 通过算法众包，建立精准的预测模型，为产品数据增长赋能
  - 探索数据人才与企业需求的生态构建

参赛流程

> PC端打开 https://www.flyai.com/?s=pmgO8ng7J 并通过微信扫码成功注册
>
> 项目列表

FlyAI提供项目涵盖图像、语音、自然语言处理及趣味性测试项目超过50例。每个项目包含数据集及来源、论文来源、数据描述介绍，达到对项目的全方面了解。

用户商场评价情感分析

奖金池 ￥ 1,000

此数据集涵盖了24 万家餐馆，54 万用户，440 万条评论/评分数据。每条数据包含用户对餐馆的四个维度的评分（总体，环境，口味，服务），评分为从1到5的整数。该数据集适合做推荐系统...

简单 自然语言处理

711

美国点评网站Yelp评价预测赛 已报名

奖金池 ￥ 12,000

该数据集包含美国15万用户对18万家企业的100万条点评，涵盖超过140万个商业属性，包括营业时间，停车位，便利性和环境等等。每条数据包括企业名称，位置，属性和类别

简单 自然语言处理 Keras

5800

人脸关键点检测(五点) 已报名

奖金池 ￥ 10,000

这些图片是用亚马逊机械土耳其人手工标注的，以精确定位眼睛、鼻子和嘴。最终的数据集包括40000个图像，其中主要面部组件的注释高度准确、详细和一致。

困难 人脸关键点检测 TensorFlow

3865

测测星座文本分类 已报名

奖金池 ￥ 1,000

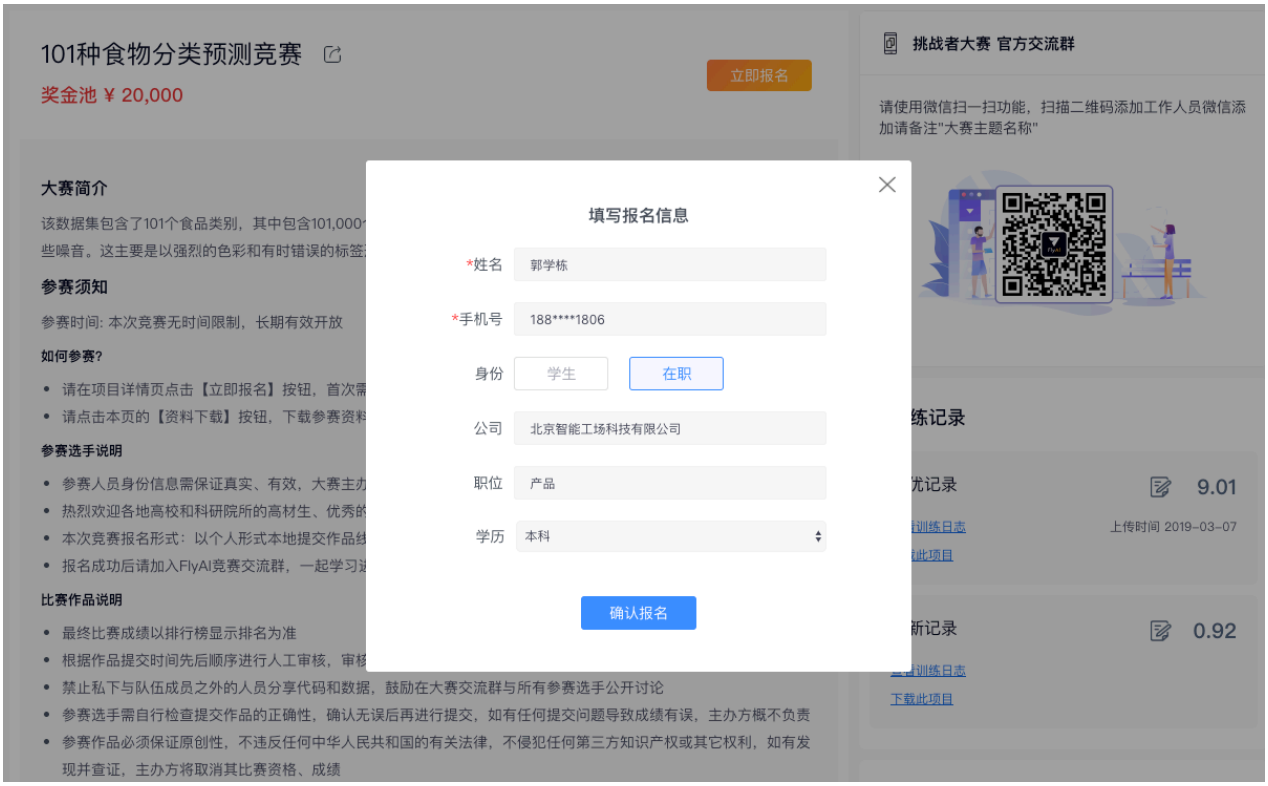
首次由测测星座提供文本分类数据，致力于发掘在当下的AI时代为娱乐产业赋能升级的最佳解决方案。目前测测已经发展成为中国最大的移动星座社区，积累了近千万的个人档案数据，构建了全...

中等 文本分类 TensorFlow

1899

立即报名

填写参赛信息



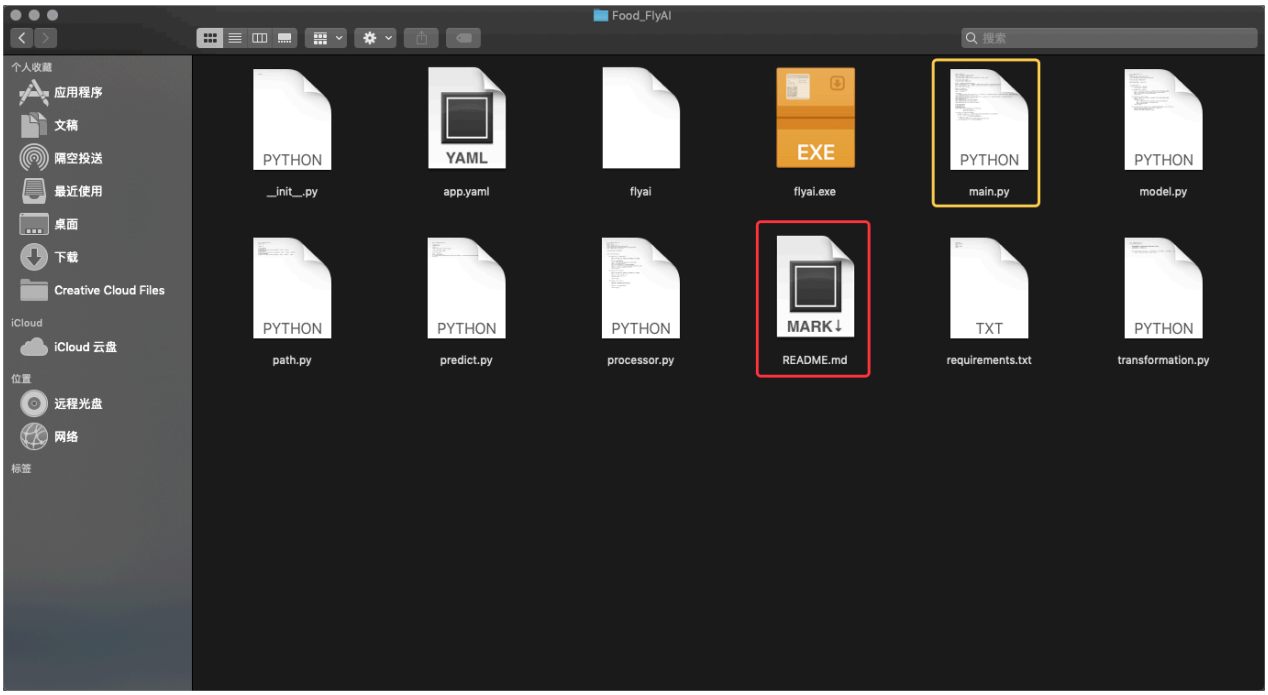
报名成功，点击下载参赛资料

报名成功!

感谢您的参与！点击 下载资料 并及时提交您的作品

下载资料

> 本地解压文件包，并打开文件，仔细阅读README.md!

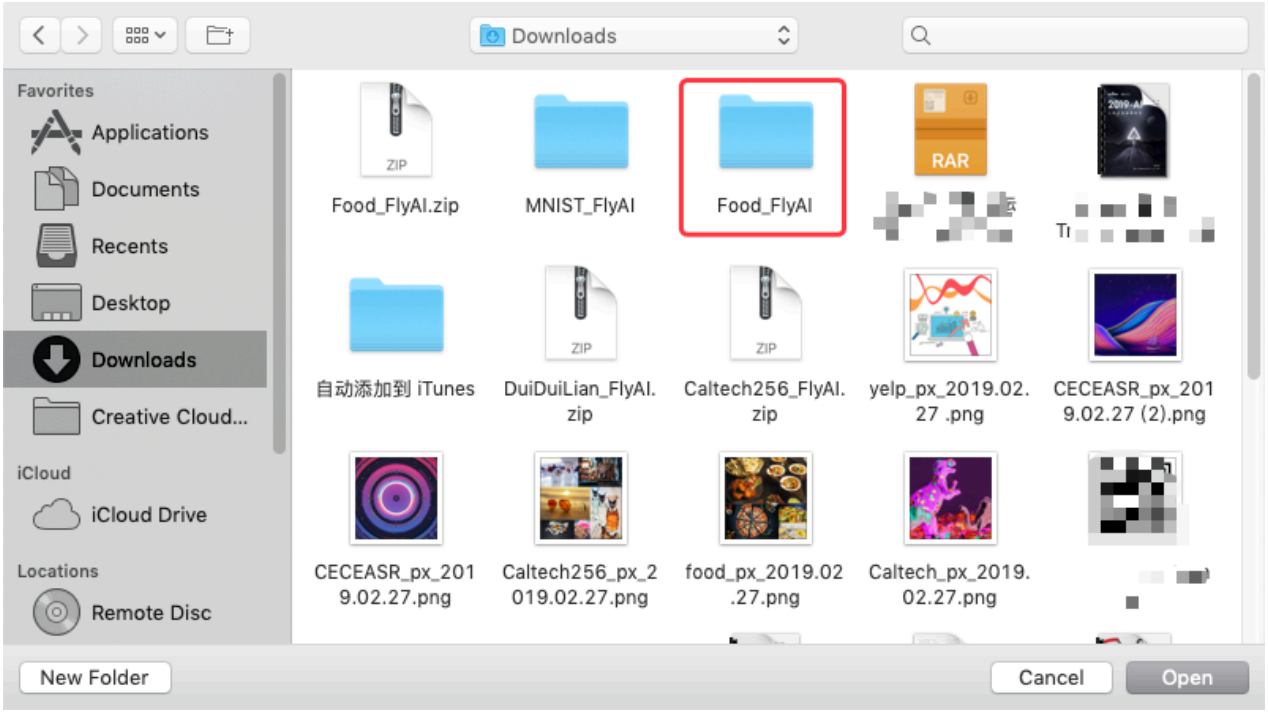


如需使用PyCharm开发，环境配置说明

> 下载PyCharm社区版

- windows安装 http://class.xxwolo.com/pycharm-community-2018.3.3.exe
- Mac安装 http://class.xxwolo.com/pycharm-community-2018.3.3.dmg

> 在 PyCharm 中 Open Food_FlyAI



> 在PyCharm初始化环境

- 为flyai脚本添加权限（windows用户跳过此步骤，进行下一步）

```
# 在PyCharm左下方点击Terminal，在命令行中输入对应命令：

# Mac
chmod +x flyai_darwin
mv flyai_darwin flyai
```

- 初始化环境

```
# 在PyCharm左下方点击Terminal，在命令行中输入对应命令

# Windows
flyai.exe init
# Mac
./flyai init
```

- 配置python环境

```
# 1. 在PyCharm中点击 File

# 2. Windows用户点击：Settings
     Mac     用户点击：Default Settings

# 3. Windows用户点击：Project {项目名称}->Project Interperter
     Mac     用户点击：Project Interperter

# 4. 点击右上方的小齿轮，然后点击 Add
```

```
# 5. 选择Existing environment

# 6. 点击Interpreter输入框后面的...

# 7. Windows用户选择：C:\Users\{你计算机用户名}\.flyai\env\python.exe
     Mac     用户选择：Users/{你计算机用户名}/.flyai/env/bin/python3.6

# 8. 然后一路点击 OK
```

- 本地运行，验证环境

```
# 在PyCharm左下方点击 Terminal ，在命令行中输入对应命令：

# Windows 输入：
flyai.exe test
# Mac     输入：
./flyai test
```

参赛资料包文件说明

`app.yaml`

> 是项目的配置文件，项目目录下**必须**存在这个文件，是项目运行的依赖。

`processor.py`

> 处理数据的输入输出文件，把通过csv文件返回的数据，处理成能让程序识别、训练的矩阵。
>
> 可以自己定义输入输出的方法名，在`app.yaml`中声明即可。

```
def input_x(self, image_path):
    '''
    参数为csv中作为输入x的一条数据，会被Dataset类中的next_batch()方法多次调用。
    :params: 输入的数据列表
    :return: 返回矩阵
    '''
    pass

def input_y(self, label):
    '''
    参数为csv中作为输入y的一条数据，会被Dataset类中的next_batch()方法多次调用。
    :params: 数据标签列表
    :return: 返回矩阵
    '''
    pass
def output_y(self, data):
    '''
    验证时使用，把模型输出的y转为对应的结果
```

```
    :param data: 预测返回的数据
    :return: 返回预测的标签
    '''
    pass
```

`main.py`

> 程序入口，编写算法，训练模型的文件。在该文件中实现自己的算法。
>
> 通过`dataset.py`中的`next_batch`方法获取训练和测试数据。

```
# 数据获取辅助类
dataset = Dataset()
x_train, y_train, x_test, y_test = dataset.next_batch(BATCH)
```

> 通过`model.py`中的`save_model`方法保存模型

```
# 模型操作辅助类
model = Model(dataset)
model.save_model(YOU_NET)
```

> **如果使用`PyTorch`框架，需要在`net.py`文件中实现网络。其它用法同上。**
>
> **样例代码中已做简单实现，可供参考:**

```
import argparse
import tensorflow as tf
import tensorflow.contrib.slim as slim
from flyai.dataset import Dataset

from model import Model
from path import MODEL_PATH, LOG_PATH

# 数据获取辅助类
dataset = Dataset()

# 模型操作辅助类
model = Model(dataset)

'''
使用tensorflow实现自己的算法

'''
parser = argparse.ArgumentParser()
parser.add_argument("-e", "--EPOCHS", default=10, type=int, help="train
epochs")
parser.add_argument("-b", "--BATCH", default=32, type=int, help="batch
size")
```

```
args = parser.parse_args()

# 定义命名空间
x = tf.placeholder(tf.float32, [None, 224, 224, 3], name='input_x')
y = tf.placeholder(tf.float32, [None, 2], name='input_y')
keep_prob = tf.placeholder(tf.float32, name='keep_prob')

x_image = tf.reshape(x, [-1, 224, 224, 3])


def vgg_19(inputs,
     num_classes=2,
     is_training=True,
     dropout_keep_prob=0.5,
     spatial_squeeze=True,
     scope='vgg_19',
     fc_conv_padding='VALID',
     global_pool=False):
with tf.variable_scope(scope, 'vgg_19', [inputs]):
  with slim.arg_scope([slim.conv2d, slim.fully_connected,
slim.max_pool2d]):
      net = slim.repeat(inputs, 2, slim.conv2d, 64, [3, 3], scope='conv1')
      net = slim.max_pool2d(net, [2, 2], scope='pool1')
      net = slim.repeat(net, 2, slim.conv2d, 128, [3, 3], scope='conv2')
      net = slim.max_pool2d(net, [2, 2], scope='pool2')
      net = slim.repeat(net, 4, slim.conv2d, 256, [3, 3], scope='conv3')
      net = slim.max_pool2d(net, [2, 2], scope='pool3')
      net = slim.repeat(net, 4, slim.conv2d, 512, [3, 3], scope='conv4')
      net = slim.max_pool2d(net, [2, 2], scope='pool4')
      net = slim.repeat(net, 4, slim.conv2d, 512, [3, 3], scope='conv5')
      net = slim.max_pool2d(net, [2, 2], scope='pool5')
      net = slim.conv2d(net, 4096, [7, 7], padding=fc_conv_padding,
scope='fc6')
      net = slim.dropout(net, dropout_keep_prob, is_training=is_training,
                         scope='dropout6')
      net = slim.conv2d(net, 4096, [1, 1], scope='fc7')
      if global_pool:
          net = tf.reduce_mean(net, [1, 2], keep_dims=True,
name='global_pool')
      if num_classes:
          net = slim.dropout(net, dropout_keep_prob,
is_training=is_training,
                             scope='dropout7')
          net = slim.conv2d(net, num_classes, [1, 1],
                            activation_fn=None,
                            normalizer_fn=None,
                            scope='fc8')
          if spatial_squeeze:
              net = tf.squeeze(net, [1, 2], name='fc8/squeezed')
```

```
        return net


g = tf.Graph()
prediction = tf.add(vgg_19(x_image), 0, name='y_conv')
loss = slim.losses.softmax_cross_entropy(prediction, y)  # 读入标签
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.0001)
train_op = slim.learning.create_train_op(loss, optimizer)  # 训练以及优化

# 求准确率:
correct_prediction = tf.equal(tf.argmax(prediction, 1), tf.argmax(y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

merged = tf.summary.merge_all()
saver = tf.train.Saver()
init = tf.global_variables_initializer()

with tf.Session() as sess:
sess.run(init)
train_writer = tf.summary.FileWriter(LOG_PATH, sess.graph)
x_train, y_train, x_test, y_test = dataset.next_batch(args.BATCH)
for i in range(args.EPOCHS):
  train_dict = {x: x_train, y: y_train, keep_prob: 0.7}
  sess.run(train_op, feed_dict=train_dict)
  losses, acc_ = sess.run([loss, accuracy], feed_dict=train_dict)
  y_convs = sess.run(prediction, feed_dict=train_dict)
  print("step:{}, loss:{}, acc:{}".format(i + 1, losses, acc_))
  model.save_model(sess, MODEL_PATH, overwrite=True)
```

model.py

样例代码中已做简单实现，可供查考。

训练好模型之后可以继承`flyai.model.base`包中的`base`重写下面三个方法实现模型的保存、验证和使用。

```
def predict(self, path, name, **data):
  '''
  使用模型
  :param path: 模型所在的路径
  :param name: 模型的名字
  :param data: 模型的输入的一个或多个参数
  :return:
  '''
  pass

def evaluate(self, path, name):
```

```
    '''
    验证模型
    :param path: 模型的路径
    :param name: 模型的名字
    :return: 返回验证的准确率
    '''
    pass

def save_model(self, network, path=MODEL_PATH, name=MODEL_NAME,
overwrite=False):
    '''
    保存模型
    :param network: 训练模型的网络
    :param path: 要保存模型的路径
    :param name: 要保存模型的名字
    :param overwrite: 是否覆盖当前模型
    :return:
    '''
    self.check(path, overwrite)
```

`predict.py`

> **样例代码中已做简单实现，可供查考。**
>
> 对训练完成的模型使用和预测。

`path.py`

> 可以设置数据文件、模型文件的存放路径。

`dataset.py`

> 该文件在**FlyAI开源库**的`flyai.dataset`包中，通过`next_batch(BATCH)`方法获得`x_train` `y_train` `x_test` `y_test`数据。
>
> FlyAI开源库可以通过`pip3 install -i https://pypi.flyai.com/simple flyai`安装。

FlyAI操作命令说明

Windows用户

1. 下载项目并解压

2.打开运行，输入cmd，打开终端

> Win+R 输入cmd

3. 使用终端进入到项目的根目录下

首先进入到项目对应的磁盘中，然后执行

```
cd path\to\project
```

4. 初始化环境并登录

下载完成之后，执行下列命令并使用微信扫码登录

```
flyai.bat init
```

登录成功之后，会自动下载运行所需环境

5. 本地开发调试

执行

```
flyai.bat test
```

安装项目所需依赖，并运行 main.py

如果使用本地IDE开发，可以自行安装 requirements.txt 中的依赖，运行 main.py 即可

6.提交训练到GPU

项目中如有新的引用，需加入到 requirements.txt 文件中

在终端下执行

```
flyai.bat train
```

返回sucess状态，代表提交离线训练成功

默认训练成功后不公开在项目排行榜中，公开项目需在提交训练时执行

```
flyai.bat train -p=1
```

执行代码示例：

```
flyai.bat train -p=1 -b=32 -e=100
```

通过执行训练命令，本次训练循环 100 次，每次训练读取的数据量为 32 ，公开提交模型

Mac或Linux用户

1. 下载项目并解压

2. 使用终端进入到项目的根目录下

```
cd /path/to/project
```

3. 初始化环境并登录

授权flyai

```
chmod +x ./flyai
```

下载完成之后，执行下列命令并使用微信扫码登录

```
./flyai init
```

登录成功之后，会自动下载运行所需环境

4. 本地开发调试

执行

> ./flyai test

安装项目所需依赖，并运行 main.py

如果使用本地IDE开发，可以自行安装 requirements.txt 中的依赖，运行 main.py 即可

5.提交训练到GPU

项目中如有新的引用，需加入到 requirements.txt 文件中

在终端下执行

> ./flyai train

返回sucess状态，代表提交离线训练成功，训练结束会以微信和邮件的形式发送结果通知

默认训练成功后不公开在项目排行榜中，公开项目需在提交训练时执行

> ./flyai train -p=1

执行代码示例：

> ./flyai train -p=1 -b=32 -e=100

通过执行训练命令，本次训练循环 100 次，每次训练读取的数据量为 32 ，公开提交模型

奖金设置

> 奖励规则

奖项设置	获奖人数	奖金额度说明
参与奖（总奖金30%）	所有人	不同得分区间获得相应的竞赛奖金
突破奖（总奖金20%）	所有人	更新排行榜得分，获取相应竞赛奖金
排名奖（总奖金50%）	前3名（冠、亚、季军）	项目上线第一周、第一个月; Time_P(周) = 0.5，Time_P(月) = 0.5；K1=0.5，K2=0.3，K3=0.2；
不同框架奖励	所有人	获得60FAI币用于GPU训练资源消耗

备注：

- 按照所有挑战者项目完成训练时间先后顺序，自动审核通过可展示在排行榜中。

- 获得奖金分为3部分：参与奖、突破奖为审核完毕实时获取的奖金，排名奖需在规定时间结束后根据排名顺序获得。
- Bouns表示为：奖金池总金额；Score表示为：模型得分；
- 【参与奖】不更新得分区间不再次获得奖励；

  各项奖金获得计算公式参考如下：

  参与奖(Participation Award)

  R表示：得分的区间系数；T表示为：按照完成训练时间顺序人数；

  100-标准分：分为5个区间系数；R1(0.02),R2(0.08),R3(0.15),R4(0.25),R5(0.5)

$$PA = 0.3 * Bouns * R * \frac{1}{2^T}$$

突破奖(Prizes)

N表示：第N次更新排行榜；Prizes_N-1表示：已发放奖金总量

$$PRIZES_N = Bouns*(\frac{Score}{100})^8 - \sum Prizes_{N-1}$$

排名奖(Ranking Award)

Time_p表示：相关截止日期的奖金发放系数；K表示：每次发放排行榜前三名的分配系数；

$$RA = 0.5 * Bouns * Time_p * K$$

评判标准

> 模型得分评判说明

评审指标

- 准确率（Accuracy）：对于给定的测试数据集，预测正确的样本数与实际总样本数之比
- True，表示预测正确的样本数数量
- Total Number of Samples，表示实际总样本数数量
- 计算公式如下：

$$Score = 100 * \frac{True}{\text{Total Number of Samples}}$$

扫描下方二维码，及时获取FlyAI最新消息，抢先体验最新功能。

北京智能工场科技有限公司

2019.03.12