目录

一． 数据特征

1.1基本数据特征

数据集共 3279 个样本，即 3279 张图片所有特征数值组成的集合，所有特征中，最终的分类标准(ad 和 nonad)数据格式为 object(对象类型)，在实际数据分析中需要将其转化为整数 int 或者浮点数 float 等 Number 类型格式。其余共 1558 个特征，其中 height(图片高度/长度)，width(图片宽度)，aratio(宽/长)为连续特征，即该特征在某个范围内所有数值都有可能取到，数据格式为 object，其余为二值特征即该特征数值只有两种可能，在本实验中为 0 或 1。在所有二值特征中，除了 local，457 个来自 url，495 个来自 origurl，472 个来自 ancurl，111 个来自 alt，其余 19 个来自 caption，数据格式均为 int。在所有样本中，2820 个为 ‘nonad’ 即正常网页 内容属性样本，其余 459 个为 ‘ad’ 即广告属性样本，比例接近 6 : 1，存在轻微不均衡的现象;其中 925 个样本共 2729 个数据为缺失值，即由于缺少信息而造成的数据不完全，分别来自特征 height，width，aratio 和 local。

1.2 初步分析

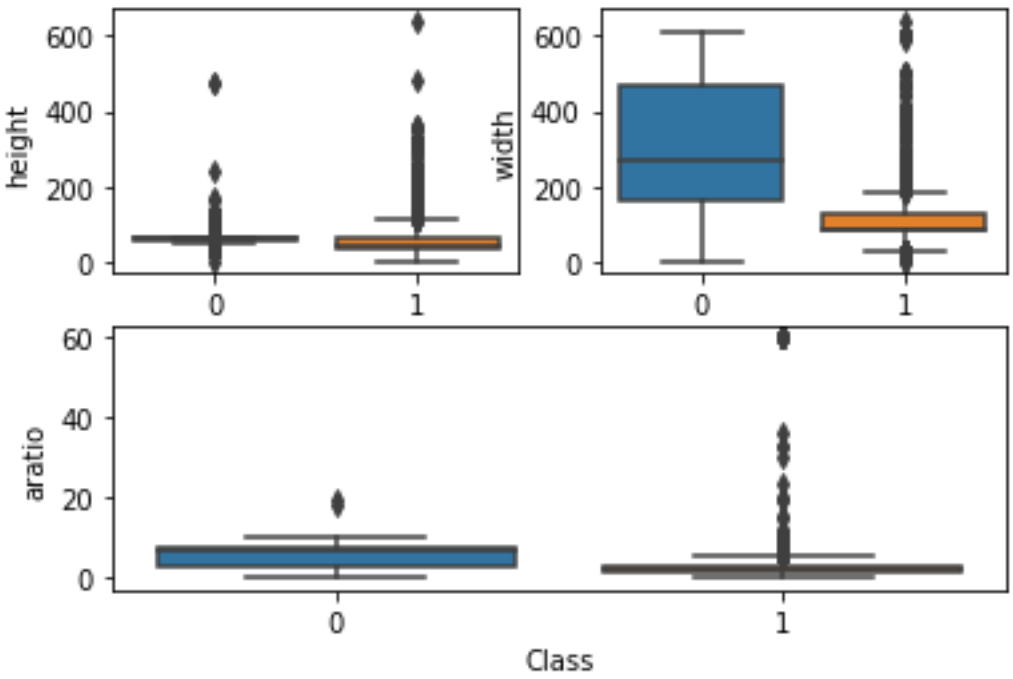

图 1.ad 和 nonad 对应 height，width，aratio 的分布情况

使用 matplotlib 工具绘出部分特征 height，width 和 aratio 进行 ad 以及 nonad 的样本分布图，我们发现广告图片和正常网页图片的高度，宽度和高宽比分布明显不同，正常网页图片的高，宽和高宽比均低于广告图片，说明分类效果明显，可用分类器进行分类。

二． 模型选择

本实验旨在建立广告分类器的同时，对贡献度高(即随着某一特征的变化，分类结果的分布也会出现明显变化)的特征进行分析及解释，所以在衡量模型预测效果的同时也需要考虑模型的可解释性。其次由于样本存在不均衡现象，所以需要均衡样本， 从而可能导致模型的预测能力下降，所以本实验将选择预测能力较高，解释性良好的 XGBoost 算法，神经网络算法以及逻辑回归算法进行分类并通过 AUC 值衡量模型预测能力，AUC 值越高，模型预测能力越好。

三． 模型定义及原理

3.1 XGBoost 算法

XGBoost 是 Boosting 算法的其中一种。Boosting 算法的思想是对样本集进行操作 获得样本子集，并通过样本子集训练出多个弱分类器，最后将许多弱分类器集成在一起 形成一个强分类器。而 XGBoost 是一种提升树模型，它是将上一棵树的残差(可理解为预测误差)作为下一棵树的分类属性，从而将许多树模型集成在一起减小误差，形成一个很强的分类器，而所用到的树模型则是 CART 分类回归树模型。CART 分类回归 树模型(Classification and Regression Tree)是一种二分递归分割技术，把当前样本划 分为两个子样本。

具体来说，设 $x_1, x_2, ..., x_n$ 代表单个样本的 n 个属性，y 表示所属类 别。

CART 算法通过递归的方式将 n 维的空间划分为不重叠的矩形，划分步骤大致如下:

(1) 选一个属性 x_i，再选取 x_i 的一个值 v_i，v_i 把 n 维空间划分为两部分，一部分的所有点都满足 $x_i \le v_i$，另一部分的所有点都满足 $x_i > v_i$，对非连续变量来说属性值的取值只有两个，即等于该值或不等于该值。对于 v_i 的选取，我们考虑去划分后的每个节点的杂质量划分所占比率之和，即 Gini 系 数，可理解为模型的混乱程度。简单来说，对于某个属性划分出的两个节点 A，B， 计算他们的加权 Gini 系数和，取 Gini 系数最小时的划分标准。

(2) 递归处理，将上面得到的两部分按步骤(1)重新选取一个属性继续划分，直到把整个 n 维空间都划分。

3.2 神经网络算法

神经网络是由一个个的被称为“神经元”的基本单位构成，单个神经元的结构如下图所示：

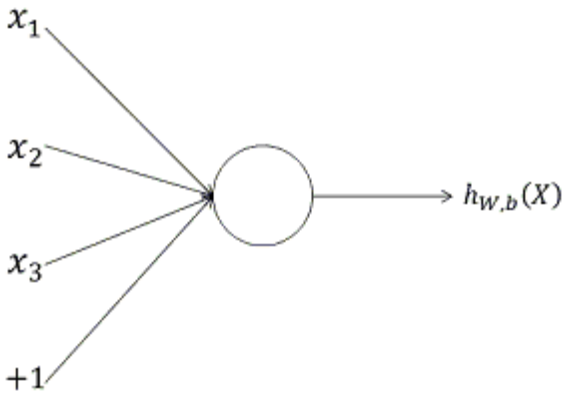

图 2. 神经元

对于上述的神经元，其输入为x_1，x_2，x_3以及截距+1，其输出为：

$$h_{W,b}(x) = f(W^T x) = f(\sum_{i=1}^{3} W_i x_i + b)$$

其中，W 表示的是向量，代表的是权重，函数 f 称为激活函数，通常激活函数可以选择为 Sigmoid 函数或者 tanh 双曲正切函数。若是使用 sigmoid 作为神经元的激活函数，则当神经元的输出为 1 时表示该神经元被激活，否则称为未被激活。同样，对于激活函数是 tanh 时，神经元的输出为 1 时表示该神经元被激活，否则称为未被激活。神经网络是由很多的神经元联结而成的，一个简单的神经网络结构如下图所示：

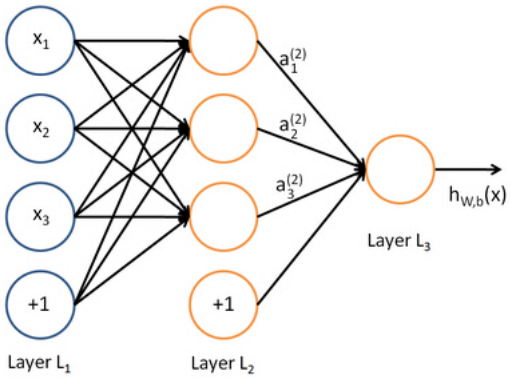

图 3. 一种神经网络结构

其中一个神经元的输出是另一个神经元的输入，+1 项表示的是偏置项。上图是含有一个隐含层的神经网络模型，L1 层称为输入层，L2 层称为隐含层，L3 层称为输出层。在神经网络中，一个神经元的输出是另一个神经元的输入。假

设z_i^l表示的是第 L 层的第 i 个神经元的输入，假设a_i^l表示的是第 L 层的第 i 个神经元的输出，其中，当 L=1 时，$a_i^l = x_i$。根据上述的神经网络中的权重和偏置，就可以计算神经网络中每一个神经元的输出，从而计算出神经网络的最终的输出$h_{W,b}$。上述的步骤称为前向传播，指的是信号从输入层，经过每一个神经元，直到输出神经元的传播过程。对于上述神经网络模型，假设有 m 个训练样本，对于一个训练样本其损失函数为：

$$J(W,b;x,y) = \frac{1}{2}\left\|h_{W,b}(x) - y\right\|^2 + R$$

其中前一项表示损失函数，后一项表示的是正则项。我们的目的是要求得参数 W 和参数 b 以使得损失函数$J(W,b)$达到最小值，首先需要对参数进行随机初始化，即使参数初始化为一个很小的接近 0 的随机数。在随机初始化参数后，利用前向传播得到预测值$h_{W,b}(x)$，进而可以得到损失函数，此时需要利用损失函数对其参数进行调整，可以利用梯度下降的方法。在计算参数的更新公式中需要用到反向传播法。总结来看，神经网络的学习过程大致如下：

（1）. 初始化参数，包括权重、偏置、网络层结构，激活函数等等；

（2）.循环计算

1.正向传播，计算误差；

2.反向传播，调整参数；

（3）.返回最终的神经网络模型

3.4 逻辑回归算法

逻辑回归算法的思想是将 $x_1, x_2, \ldots, x_n$ 这 n 个属性和所属类别 y 拟合成普通线性回归模型 $y = W^T x + b$（可理解为离所有样本距离之和最小的直线），再通过 sigmoid 函数$g(y) = \frac{1}{1+e^{-y}}$将 y 转换成落在 0−1 之间的函数，最后通过是否大于 0.5 进行分类。

四． 模型参数定义

4.1 XGBoost 参数

(1).learning_rate:即学习的速率，默认 0.1，数值越大，学习速率越快，本实验设置测试参数值为［0.05，0.06，0.07，0.08，0.09，0.1］

(2).gamma:即用来比较每次节点分裂带来的收益，默认 0，有效控制节点的过度分裂，本实验设置测试参数值为［0，0.1，0.2，0.3，0.4，0.5，0.6，0.7，0.8，0.9，1.0］

(3).max_depth:树的最大深度，max_depth 越大，模型会学到更具体更局部的样本，本实验测试参数值为［2, 3, 4, 5, 6, 7, 8, 9, 10］

4.2 神经网络参数

（1）. activation :激活函数,{‘identity’, ‘logistic’, ‘tanh’, ‘relu’}, 默认‘relu’。

1.identity: f(x) = x

2.logistic: 其实就是 sigmod,f(x) = 1 / (1 + exp(-x)).

3.tanh: f(x) = tanh(x).

4.relu: f(x) = max(0, x)

（2）. solver: {‘lbfgs’, ‘sgd’, ‘adam’}, 默认 adam，用来优化权重。

1.lbfgs: quasi-Newton 方法的优化器

2.sgd: 随机梯度下降

3.adam: Kingma, Diederik, and Jimmy Ba 提出的机遇随机梯度的优化器

默认 solver ‘adam’在相对较大的数据集上效果比较好（几千个样本或者更多），对小数据集来说，lbfgs 收敛更快效果也更好。

（3）. batch_size : int , 可选的，默认‘auto’,随机优化的 minibatches 的大小 batch_size=min(200,n_samples)，如果 solver 是‘lbfgs’，分类器将不使用 minibatch，本实验测试参数值为［100，110，120，130，140，150，160，170，180，190，200］。

（4）. max_iter: int，可选，默认 200，最大迭代次数。

（5）. learning_rate:学习率，可选 constant,invscaling,adaptive。其中 constant 是初始学习率给定的恒定学习率，invscaling 使用逆缩放指数在每个时间逐渐降低学习速率，adaptive 将初始学习率保持直至每两个连续时期未能将训练损失减少至阀值，则将当前学习率除以 5。

4.3 逻辑回归参数

(1).C(惩罚系数):正则化系数 λ 的倒数，默认 1。C 越大，说明越不能容忍出现误差，容易过拟合，导致预测精度不佳;C 越小，容易欠拟合，导致预测精

度不佳，本实验测试参数值为 [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]

(2).class_weight(分类权重):用于标示分类模型中各种类型的权重，默认为 None， 也就是不考虑权重。可以选择输入 balanced 或自行输入权重进行均衡样本。比如对于分类属性为 0 和 1 的模型，我们可以选择 balanced，那么类库会根据训练样本量来计算权重，某种类型样本量越多，则权重越低，样本量越少，则权重越高。 或者可以定义 class_weight={0 : 0.9, 1 : 0.1}，这样类型 0 的权重为 90%，而类型 1 的权重为 10%,本实验选择为 balanced。

(3).penalty: 惩罚项，可选参数为l_1, l_2。用于指定惩罚项中使用的规范，l_1规范假设的是模型的参数满足拉普拉斯分布，l_2假设的模型参数满足高斯分布，所谓的范式就是加上对参数的约束，使得模型更不会过拟合。本实验采用默认值l_2

(4).solver: solver：逻辑回归损失函数的优化方法，有四种算法供选择：

‘newton-cg’：坐标轴下降法来迭代优化损失函数

‘lbfgs’：, ‘liblinear’：牛顿法变种

‘sag’：随机梯度下降

其中‘newton-cg’, ‘lbfgs’, ‘sag’只适用于 L2 惩罚项的优化，liblinear 两种都适用。因为 L1 正则化的损失函数不是连续可导的，而{‘newton-cg’, ‘lbfgs’,‘sag’}这三种优化算法时都需要损失函数的一阶或者二阶连续导数。而‘liblinear’并没有这个依赖。当样本数目比较大时，使用 sag 效果较好，因为它只使用一部分样本进行训练。

五． 训练过程

数据预处理：
（1）. 将 features 名称所在文件读取进来，注意这里编码不是‘utf-8’，二是‘gbk’。将文件中的名称一行一行读取并存到列表中。
（2）.首先数据皆为 txt 格式，将数据用 pandas 进行读取，将列名 names 设置为上述的 features 列表，观察数据并发现其中有很多‘？’。
（3）.将问号用 np.where 方法找到‘？’并更改为空值（nan），这里遇到的问题是总是替换不成功；经过用 unique 方法进行查看，发现很多‘？’前面都存在长度不一的空格，经更改后解决问题。
（4）.对于缺失值的处理这里用其所对应类别的均值来替换
（5）.由于这里样本不均衡，使用 imblearn 包中的过抽样方法 SMOTE 对数据样本进行均衡，这里的问题是，对于逻辑回归分类器，使用 SMOTE 的效果并不好

（不如调参时将‘class_weight’设置为‘balanced’），本实验选用使分类器效果最好的均衡样本方法。
（6）.然后对数据进行归一化处理
（7）.对数据进行 PCA 降维

分类器建立：
（1）.创建各个模型分类器，设置名字和需要测试的参数
（2）. 通过训练集数据拟合模型，完成分类器的建立

最优参数选择：
通过 GridsearchCV 选择使模型准确率最高的参数并返回准确率以便进行对比

可视化分析：
（1）. 通过折线图针对模型参数的相应的 AUC 值进行比较
（2）. 通过柱状图进行特征贡献度排名

六． 结果展示

	XGBoost	神经网络	逻辑回归
AUC 值	0.9771	0.9627	0.9531

表 1.各分类器准确率对比

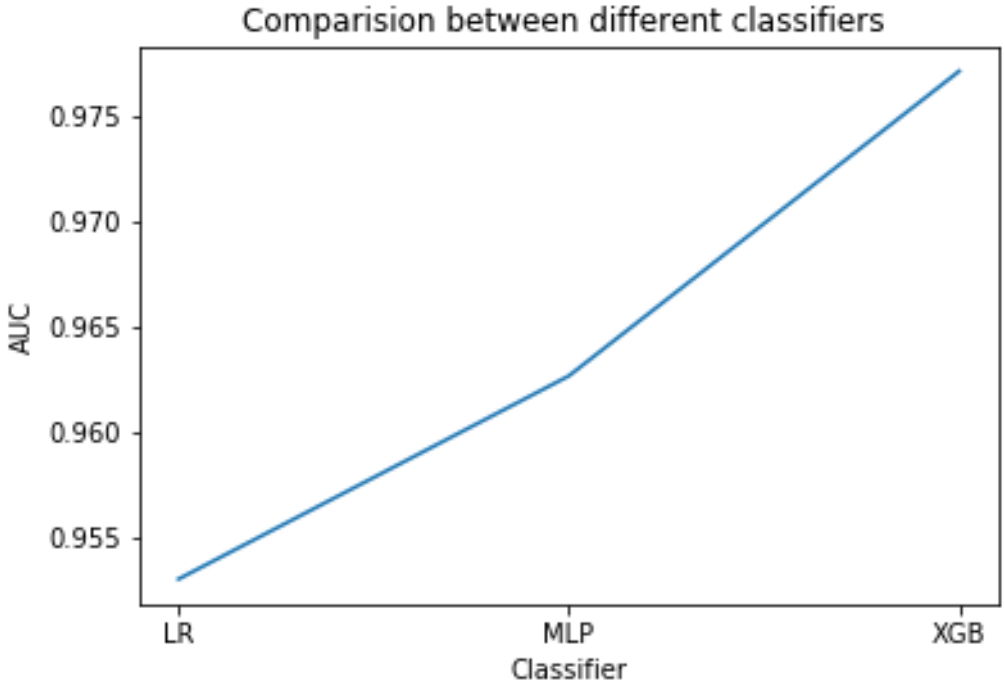

图 4.各分类器准确率对比图

从图中可以看出，XGBoost 的分类 AUC 值最高；逻辑回归的分类 AUC 值最低，下面将结合各分类器的特征贡献度进行分析。

七． 特征重要度排名

7.1 XGBoost

7.1.1 SHAP 模型

SHAP 模型是 XGBoost 算法常用特征贡献度分析模型，它将博弈论与局部解释联 系起来，根据期望表示唯一可能的一致和局部精确的加性特征归属方法。SHAP 将模型 的预测值解释为每个输入特征的归因值之和 (1 ≤ m ≤ M)，p 是输入特征的数目(本实验中为 31)， 是每个特征的归因值(Shapley 值)，是解释模型的常数，相当于训练样本结果的平均值(在本实验分类树的情况下，则为 ‘Class’ = 1 的比率);而是含有某一特 征的所有子集合的条件期望，也就是根据下式进行计算：

$$\phi_m = \sum_{S \subset \frac{\{X_1,\ X_2,\ X_3,\ \ldots,\ X_p\}}{\{X_m\}}} \frac{|S|!\,(p-|S|-1)!}{p!}\left(f_x(S \cup \{x_m\}) - f_x(S)\right)$$

其中$\{X_1,\ X_2,\ X_3,\ \ldots,\ X_p\}$是输入特征的集合，$f_x(S)$为特征子集 S 的预测均值，而$\frac{|S|!(p-|S|-1)!}{p!}$为权重，因为在确定子集$S$后，$p$个特征在特定排序的情况下有$|S|!\,(p-|S|-1)!$种组合情况。简单来说，若某一特征的 SHAP 值大于 0，表示该特征对分类结果产生正影响， 即该特征值越大，将交易分类为欺诈 (‘Class’ = 1)的概率越大，反之若某一特征的 SHAP 值小于 0，表示该特征对分类结果产生负影响，即该特征值越大，将交易分类为非欺诈 (‘Class’ = 0)的概率越大;而某一特征的 SHAP 值的绝对值越大，则该属性对分类结果的影响越明显，对模型的贡献度越高。

7.1.2 特征贡献度

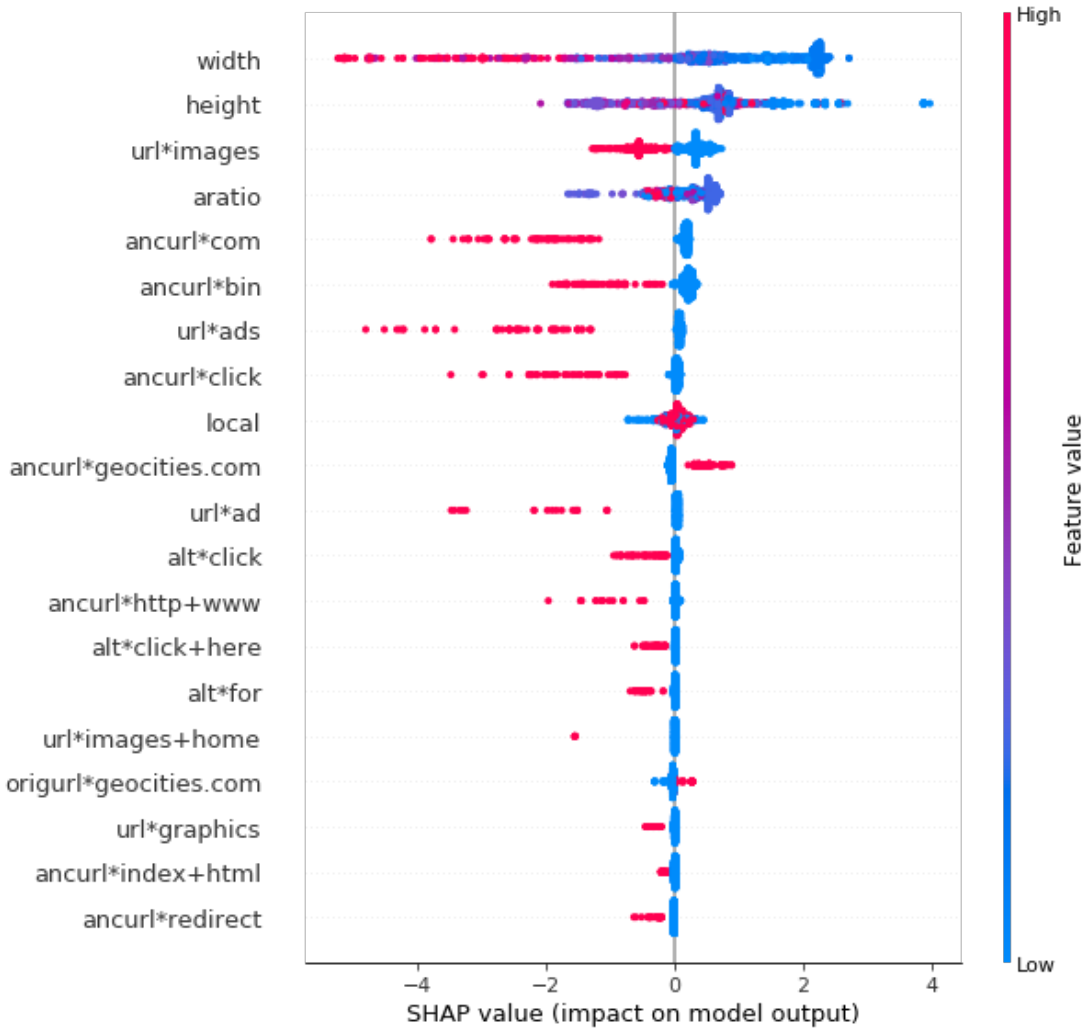

图 5. XGBoost 特征贡献度

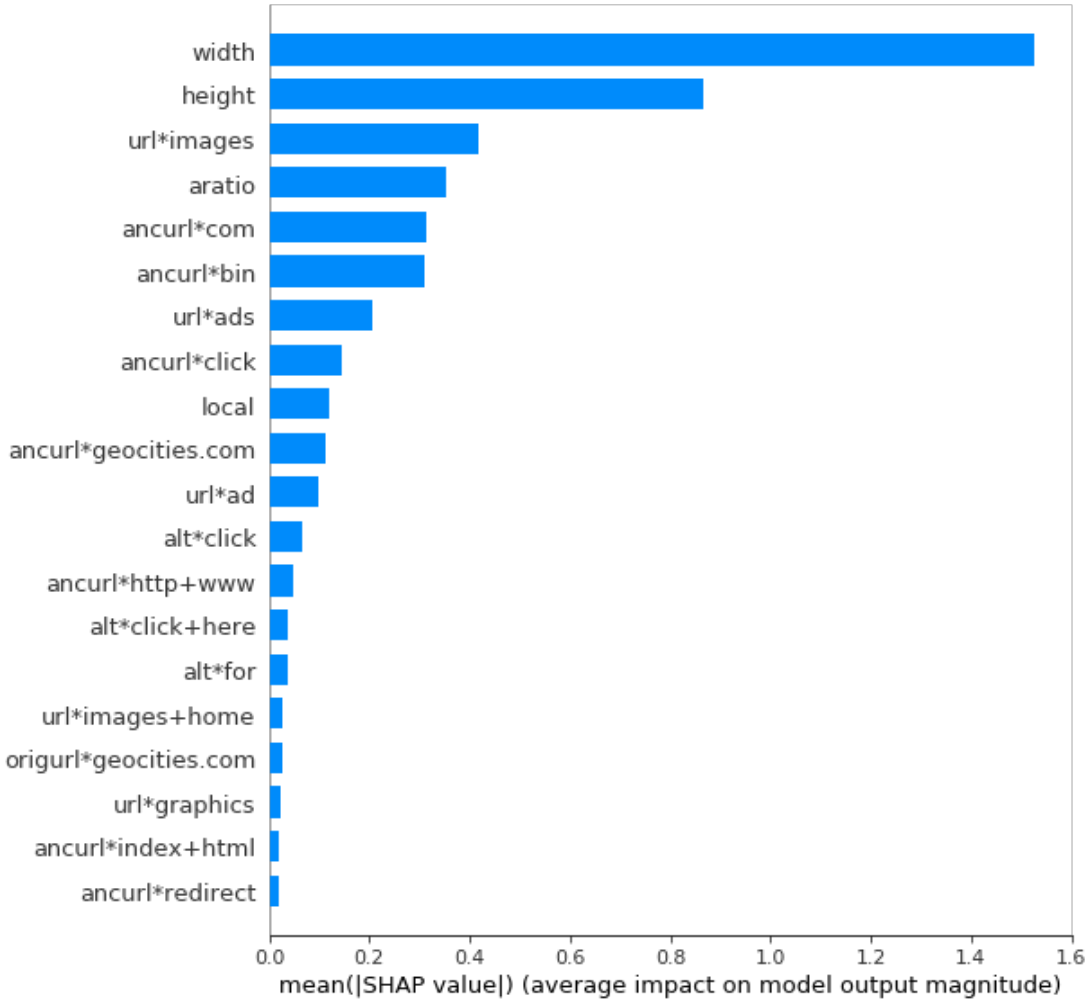

图 6. XGBoost 特征贡献度

7.2 神经网络

7.2.1 置换重要性

置换法的思路是：我们可以通过当某一个特征不存在时观察一个模型准确度的改变情况来衡量该特征的特征重要性。要做到这一点，可以从数据集中删除特征，重新训练估计器并检查得分。但它需要为每个特征重新训练一个估计器，这可能是需要大量计算的。另外，它显示的是数据集内可能重要的东西，而不是具体的训练模型内重要的东西。为了避免重新训练估计器，我们可以只从数据集的测试部分删除一个特征，然后在不使用这个特征的情况下计算得分。但是估计器期望特征是存在的。因此，我们可以用随机噪声代替删除特征--特征列仍然存在，但它不再包含有用的信息。如果噪声与原始特征值的分布相同，这种方法就可以工作（因为否则估计器可能会失败）。获得这种噪声最简单的方法是对特征值进行洗牌。本实验我们使用的是 eli5 库中的 PermutationImportance 方法，一种通过测量当一个特征不可用时准确度如何下降来计算任何黑盒估计器的特征贡献度的方法。

（https://eli5.readthedocs.io/en/latest/blackbox/permutation_importance.html#eli5-permutation-importance）

7.2.2 特征贡献度

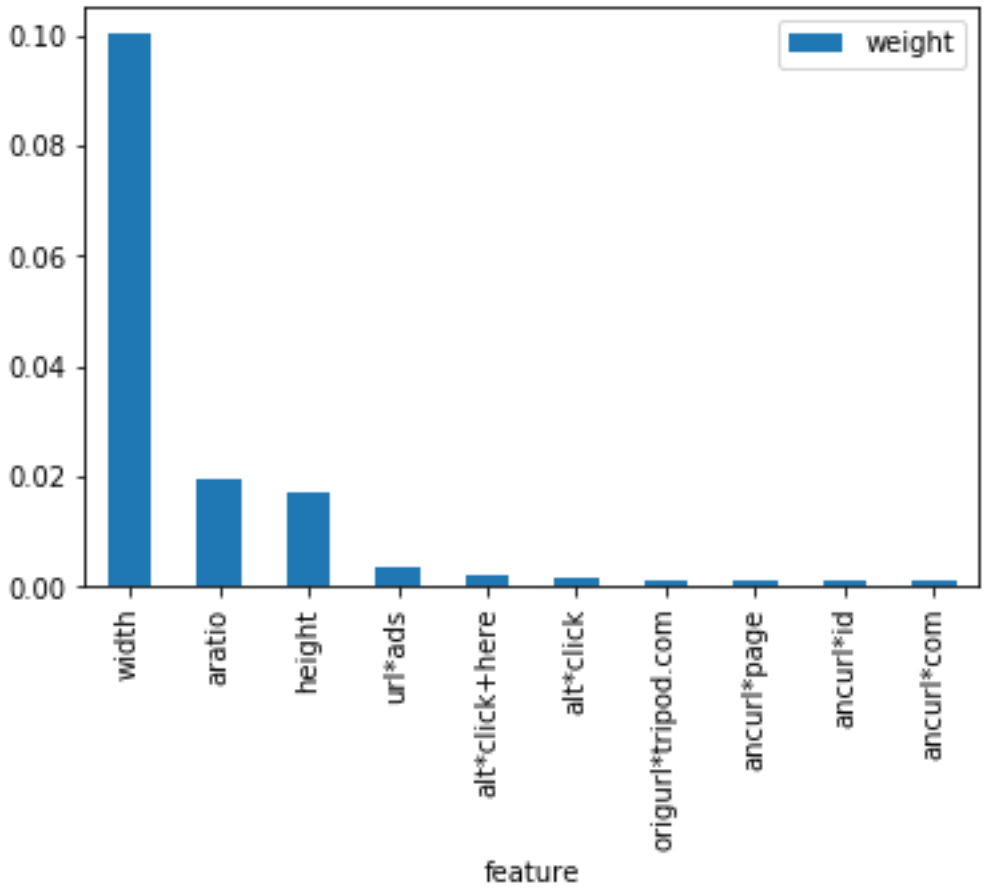

图 7. 神经网络特征重要度

7.4 逻辑回归

由于逻辑回归算法的思想是将X_1，X_2，X_3，...，X_n 这 n 个属性和所属类别 y 拟合成普通线性回归模型$y = \beta_0 + \sum_{m=1}^{n} \beta_m x_m$,所以我们可以直接用 β_m 来分析特征贡献度，简单来说，若某一特征 x_m的系数 β_m 大于 0，表示该特征对分类结果产生正影响，即该特征值越大，将交易分类为欺诈（‘Class’=1）的概率越大，反之若某一特征 x_m 的系数 β_m 小于 0，表示该特征对分类结果产生负影响，即该特征值越大，将交易分类为非欺诈(‘Class’ = 0)的概率越大;而若某一特征x_m的系数 β_m 的绝对值越大，则该属性对分类结果的影响越明显，对模型的贡献度越高。下面是本实验得到的特征重要度的排名：

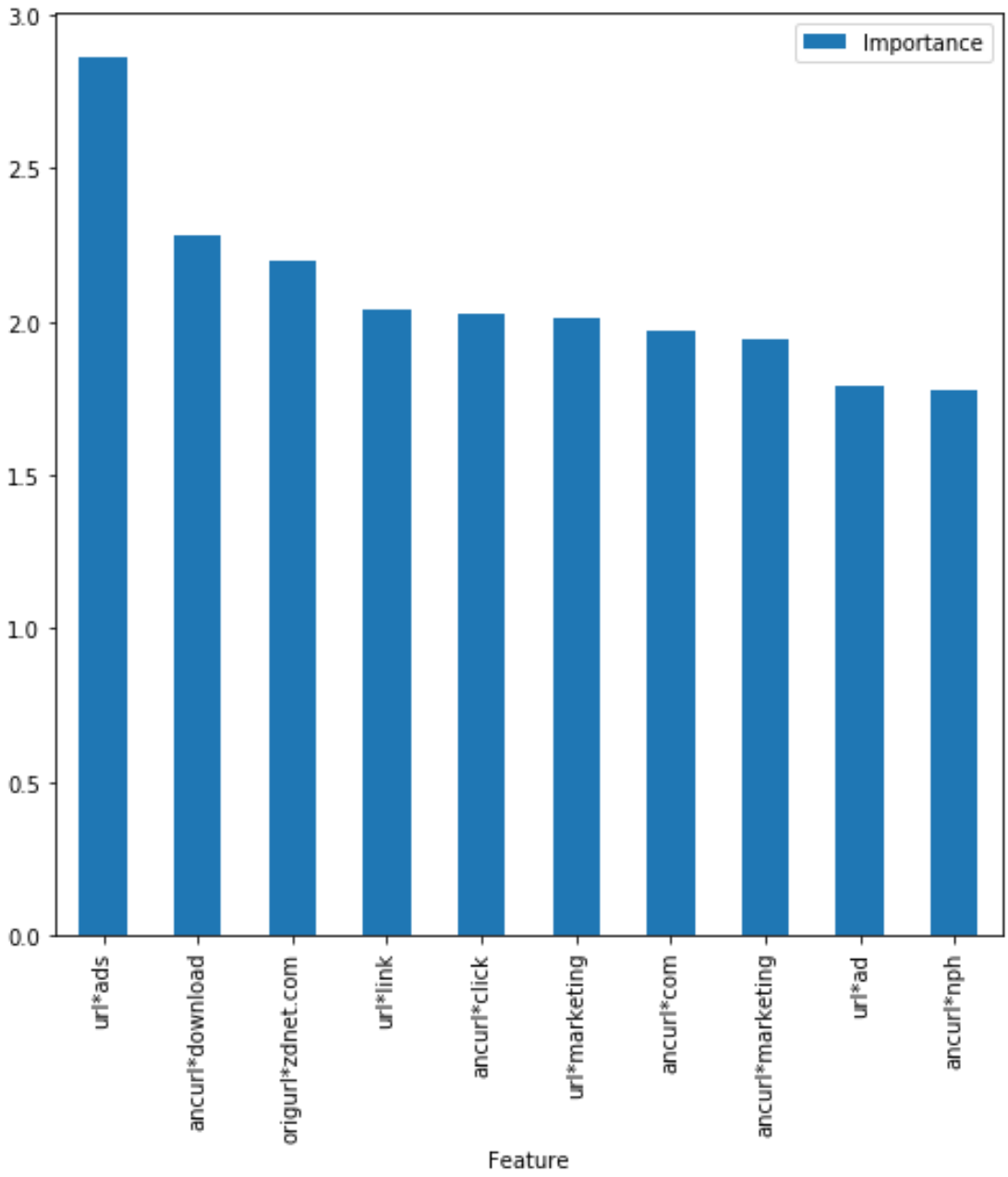

图 8. 逻辑回归特征贡献度

综合所有特征贡献度的图可以看出，对分类贡献最大的特征主要是‘width’，‘height’，‘aratio’等连续特征，其次是 url*ads。