《机器学习》

课程设计报告

基于决策树对年收入的分类

**学院：信息科学技术学院**

**班级：数据二班**

**学号：202044514210**

**姓名：林长坤**

**指导老师：陈付郴**

**完成日期：2023年6月**

**目录**

[一、 选题分析 3](#_Toc21760)

[1.1 问题分析 3](#_Toc30464)

[1.2 求解目标 3](#_Toc1691)

[二、 数据准备 4](#_Toc26342)

[2.1 探索数据 4](#_Toc2266)

[2.2 数据预处理 4](#_Toc18928)

[2.2.1 获得特征和标签 4](#_Toc910)

[2.2.2 转换倾斜的连续特征 4](#_Toc19209)

[2.2.3 归一化数字特征 6](#_Toc5828)

[2.2.3 独热编码转换类别变量 6](#_Toc16569)

[2.2.3 切分数据 7](#_Toc26324)

[三、 研究过程 7](#_Toc7725)

[3.1 评价性能指标 7](#_Toc10967)

[3.2 监督学习模型 7](#_Toc10257)

[3.2.1 构造训练器 8](#_Toc6334)

[3.2.2 数据划分 8](#_Toc11655)

[3.2.3 模型训练 8](#_Toc14002)

[3.3 参数优化 8](#_Toc4304)

[3.4 特征重要性与特征训练 8](#_Toc32005)

[四、 总结与展望 9](#_Toc30311)

[五、 参考文献 9](#_Toc1840)

[五、附录 9](#_Toc31038)

# 选题分析

## 问题分析

在这个项目中，使用1994年美国人口普查收集的数据，选用几个监督学习算法以准确地建模被调查者的收入。然后，根据初步结果从中选择出最佳的候选算法，并进一步优化该算法以最好地建模这些数据。

了解人群的收入情况可以帮助一个非营利性的机构更好地了解他们能得到多大的捐赠，或是否他们应该接触这些人。虽然很难直接从公开的资源中推断出一个人的一般收入阶层，但是可以从其他的一些公开的可获得的资源中获得一些特征从而推断出该值。

## 求解目标

目标是建立一个能够准确地预测被调查者年收入是否超过50000美元的模型。这种类型的任务会出现在那些依赖于捐款而存在的非营利性组织。

表1 特征数据集

|  |  |  |  |
| --- | --- | --- | --- |
| age | workclass | education\_level | education-num |
| 年龄 | 劳动类型 | 教育程度 | 学校学习时间 |
| marital-status | occupation | relationship | race |
| 婚姻状况 | 职业领域 | 家庭情况 | 人种 |
| sex | capital-gain | capital-loss | hours-per-week |
| 性别 | 资本收益 | 资本损失 | 工作时间 |
| native-country | income |  |  |
| 国家 | 收入类别 |  |  |

人口普查数据集含有45222个数据点，共含有13个属性集。

# 数据准备

## 2.1 探索数据

首先对数据集进行一个粗略的探索，我们将看看每一个类别里会有多少被调查者？并且告诉我们这些里面多大比例是年收入大于50,000美元的。

|  |  |  |  |
| --- | --- | --- | --- |
| 年收入 | 大于50000 | 小于50000 | 占比 |
| 人数 | 11208 | 34014 | 24.78% |

## 2.2 数据预处理

在数据能够被作为输入提供给机器学习算法之前，它经常需要被清洗，格式化或重新组织----这通常被叫做预处理。幸运的是，对于这个数据集，没有我们必须处理的无效或丢失的条目，然而，由于某一些特征存在的特性我们必须进行一定的调整。这些预处理可以极大地帮助我们提升几乎所有的学习算法的结果和预测能力。

### 2.2.1 获得特征和标签

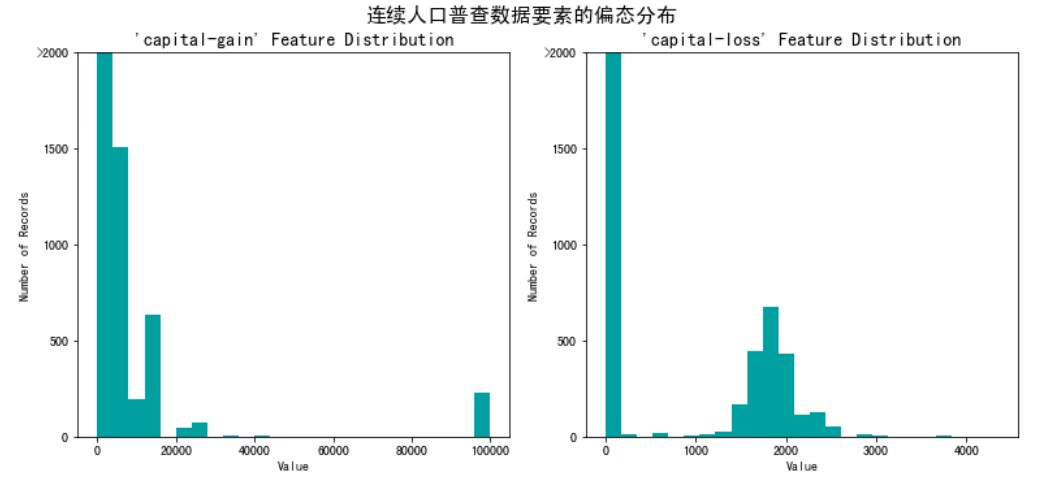
income 列是我们需要的标签，记录一个人的年收入是否高于50K。 因此我们应该把他从数据中剥离出来，单独存放。

|  |
| --- |
| # 将数据切分成特征和对应的标签  income\_raw = data['income']  features\_raw = data.drop('income', axis = 1)  # 显示分离之后的特征数据和标签数据  display(features\_raw.head())  display(income\_raw.head()) |

### 2.2.2 转换倾斜的连续特征

一个数据集可能包含至少一个靠近某个数字的特征，但有时也会有一些相对来说存在极大值或者极小值的不平衡分布的的特征。算法对这种分布的数据会十分敏感，并且如果这种数据没有能够很好地规一化处理会使得算法表现不佳。在人口普查数据集中的两个特征符合这个描述：'capital-gain'和'capital-loss'。

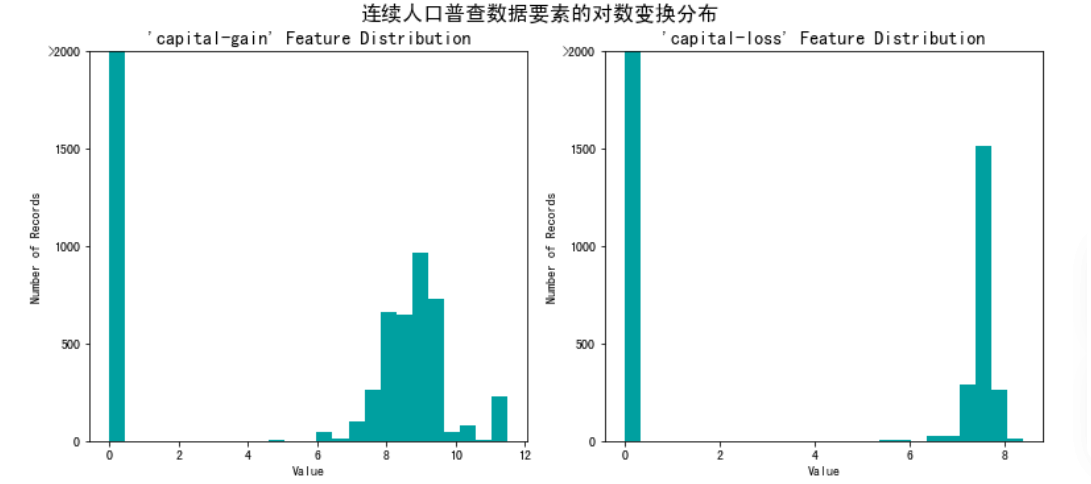
|  |
| --- |
| # 可视化 'capital-gain'和'capital-loss' 两个特征  vs.distribution(features\_raw) |



对于高度倾斜分布的特征如'capital-gain'和'capital-loss'，常见的做法是对数据施加一个对数转换，将数据转换成对数，这样非常大和非常小的值不会对学习算法产生负面的影响。并且使用对数变换显著降低了由于异常值所造成的数据范围异常。但是在应用这个变换时必须小心：因为0的对数是没有定义的，所以我们必须先将数据处理成一个比0稍微大一点的数以成功完成对数转换。

运行下面的代码单元来执行数据的转换和可视化结果。再次，注意值的范围和它们是如何分布的。

|  |
| --- |
| # 对于倾斜的数据使用Log转换  skewed = ['capital-gain', 'capital-loss']  features\_raw[skewed] = data[skewed].apply(lambda x: np.log(x + 1))  # 可视化对数转换后 'capital-gain'和'capital-loss' 两个特征  vs.distribution(features\_raw, transformed = True) |



### 2.2.3 归一化数字特征

## 除了对于高度倾斜的特征施加转换，对数值特征施加一些形式的缩放通常会是一个好的习惯。在数据上面施加一个缩放并不会改变数据分布的形式（比如上面说的'capital-gain' or 'capital-loss'）；但是，规一化保证了每一个特征在使用监督学习器的时候能够被平等的对待。注意一旦使用了缩放，观察数据的原始形式不再具有它本来的意义了，就像下面的例子展示的。

## 运行下面的代码单元来规一化每一个数字特征。我们将使用sklearn.preprocessing.MinMaxScaler来完成这个任务。

|  |
| --- |
| from sklearn.preprocessing import MinMaxScaler  # 初始化一个 scaler，并将它施加到特征上  scaler = MinMaxScaler()  numerical = ['age', 'education-num', 'capital-gain', 'capital-loss', 'hours-per-week']  features\_raw[numerical] = scaler.fit\_transform(data[numerical])  # 显示一个经过缩放的样例记录  display(features\_raw.head(n = 1)) |

### 2.2.3 独热编码转换类别变量

从上面的数据探索中的表中，我们可以看到有几个属性的每一条记录都是非数字的。通常情况下，学习算法期望输入是数字的，这要求非数字的特征（称为类别变量）被转换。转换类别变量的一种流行的方法是使用独热编码方案。独热编码为每一个非数字特征的每一个可能的类别创建一个“虚拟”变量。例如，假设someFeature有三个可能的取值A，B或者C。我们将把这个特征编码成someFeature\_A, someFeature\_B和someFeature\_C.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 特征**X** |  | 特征**X\_A** | 特征**X\_B** | 特征**X\_C** |
| B |  | 0 | 1 | 0 |
| C | --> 独热编码 --> | 0 | 0 | 1 |
| A |  | 1 | 0 | 0 |

此外，对于非数字的特征，我们需要将非数字的标签'income'转换成数值以保证学习算法能够正常工作。因为这个标签只有两种可能的类别（"<=50K"和">50K"），我们不必要使用独热编码，可以直接将他们编码分别成两个类0和1，在下面的代码单元中你将实现以下功能：

使用pandas.get\_dummies()对'features\_raw'数据来施加一个独热编码。

将目标标签'income\_raw'转换成数字项。

将"<=50K"转换成0；将">50K"转换成1。

|  |
| --- |
| import pandas as pd  # 使用pandas.get\_dummies()对'features\_raw'数据进行独热编码  features = pd.get\_dummies(features\_raw)  # 将'income\_raw'编码成数字值  #np.array([1 if value =='>50K' else 0 for value in income\_raw])  income = income\_raw.apply(lambda x: int(x==">50K"))  # 打印经过独热编码之后的特征数量  encoded = list(features.columns)  print ("{} total features after one-hot encoding.".format(len(encoded))) |

### 2.2.3 切分数据

现在所有的 类别变量 已被转换成数值特征，而且所有的数值特征已被规一化。和我们一般情况下做的一样，我们现在将数据（包括特征和它们的标签）切分成训练和测试集。其中80%的数据将用于训练和20%的数据用于测试。然后再进一步把训练数据分为训练集和验证集，用来选择和优化模型。

运行下面的代码单元来完成切分。

|  |
| --- |
| # 导入 train\_test\_split  from sklearn.model\_selection import train\_test\_split  # 将'features'和'income'数据切分成训练集和测试集  X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, income, test\_size = 0.2, random\_state = 0,  stratify = income)  # 将'X\_train'和'y\_train'进一步切分为训练集和验证集  X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_train, y\_train, test\_size=0.2, random\_state=0, stratify = y\_train)  # 显示切分的结果  print ("Training set has {} samples.".format(X\_train.shape[0]))  print ("Validation set has {} samples.".format(X\_val.shape[0]))  print ("Testing set has {} samples.".format(X\_test.shape[0])) |

# 研究过程

## 3.1 评价性能指标

## 使用准确率作为评价模型的标准是合适的。另外，把没有收入大于$50,000的人识别成年收入大于$50,000对于CharityML来说是有害的，因为他想要找到的是有意愿捐款的用户。这样，我们期望的模型具有准确预测那些能够年收入大于$50,000的能力比模型去查全这些被调查者更重要。我们能够使用F-beta score作为评价指标，这样能够同时考虑查准率和查全率：



尤其是，当β=0.5的时候更多的强调查准率，这叫做F0.5score（或者为了简单叫做F-score）。

## 3.2 监督学习模型

* 决策树：易于实现和理解。能处理多种不同的数据类型。计算消耗相对较低,结果输出容易理解。能有较高的准确率。能处理好非线性问题。
* 高斯朴素贝叶斯：推理过程很简单,训练时间短；能适应数据中含有缺失值的情况；能处理好非线性问题；可以计算出最终类别的概率。
* SVM(支持向量机)：对数据有很好的学习能力,能够适应新数据；对高纬度的数据能进行很好的处理；能很好的处理非线性问题,通过使用核函数。

### 3.2.1 构造训练器

为了正确评估你选择的每一个模型的性能，创建一个能够帮助你快速有效地使用不同大小的训练集并在验证集上做预测的训练和验证的流水线是十分重要的。 你在这里实现的功能将会在接下来的部分中被用到。在下面的代码单元中，你将实现以下功能：

* 从sklearn.metrics中导入fbeta\_score和accuracy\_score。
* 用训练集拟合学习器，并记录训练时间。
* 对训练集的前300个数据点和验证集进行预测并记录预测时间。
* 计算预测训练集的前300个数据点的准确率和F-score。
* 计算预测验证集的准确率和F-score。

|  |
| --- |
| # TODO：从sklearn中导入两个评价指标 - fbeta\_score和accuracy\_score  from sklearn.metrics import fbeta\_score, accuracy\_score  def train\_predict(learner, sample\_size, X\_train, y\_train, X\_val, y\_val):  '''  inputs:  - learner: the learning algorithm to be trained and predicted on  - sample\_size: the size of samples (number) to be drawn from training set  - X\_train: features training set  - y\_train: income training set  - X\_val: features validation set  - y\_val: income validation set  '''  results = {}  # TODO：使用sample\_size大小的训练数据来拟合学习器  # TODO: Fit the learner to the training data using slicing with 'sample\_size'  start = time() # 获得程序开始时间  learner = learner.fit(X\_train[:sample\_size],y\_train[:sample\_size])  end = time() # 获得程序结束时间  # TODO：计算训练时间  results['train\_time'] = end - start  # TODO: 得到在验证集上的预测值  #然后得到对前300个训练数据的预测结果  start = time() # 获得程序开始时间  predictions\_val = learner.predict(X\_val)  predictions\_train = learner.predict(X\_train[:300])  end = time() # 获得程序结束时间  # TODO：计算预测用时  results['pred\_time'] = end - start  # TODO：计算在最前面的300个训练数据的准确率  results['acc\_train'] = accuracy\_score(y\_train[:300],predictions\_train)  # TODO：计算在验证上的准确率  results['acc\_val'] = accuracy\_score(y\_val,predictions\_val)  # TODO：计算在最前面300个训练数据上的F-score  results['f\_train'] = fbeta\_score(y\_train[:300],predictions\_train,beta=0.5)  # TODO：计算验证集上的F-score  results['f\_val'] = fbeta\_score(y\_val,predictions\_val,beta=0.5)  # 成功  print ("{} trained on {} samples.".format(learner.\_\_class\_\_.\_\_name\_\_, sample\_size))  # 返回结果  return results |

### 3.2.2 数据划分

计算1%， 10%， 100%的训练数据分别对应多少个数据点，并将这些值存储在'samples\_1', 'samples\_10', 'samples\_100'中。

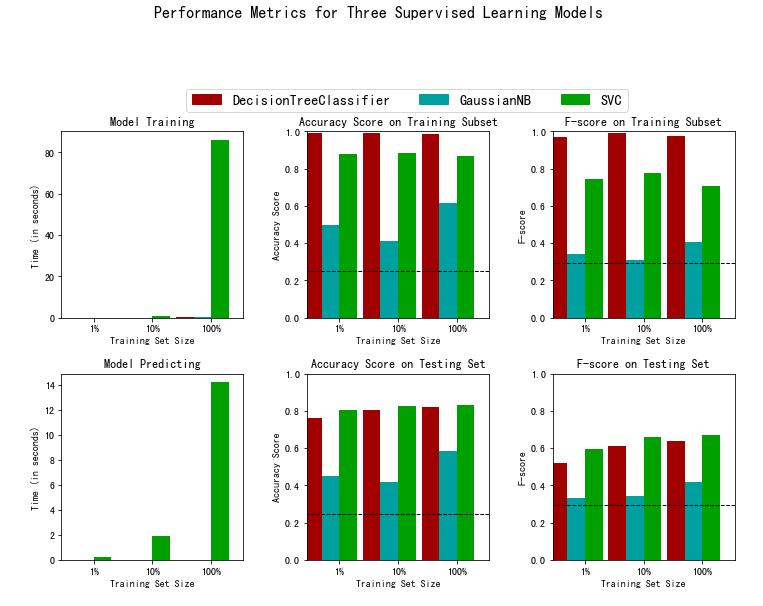
|  |
| --- |
| # TODO：计算1%， 10%， 100%的训练数据分别对应多少点  samples\_1 = int(len(X\_train) \* 0.01)  samples\_10 = int(len(X\_train) \* 0.1)  samples\_100 = len(X\_train) |

### 3.2.3 模型训练

在下面的代码单元中，您将需要实现以下功能：

* 导入你在前面讨论的三个监督学习模型。
* 初始化三个模型并存储在'clf\_A'，'clf\_B'和'clf\_C'中。
* 使用模型的默认参数值，在接下来的部分中你将需要对某一个模型的参数进行调整。
* 设置random\_state (如果有这个参数)。

|  |
| --- |
| # TODO：从sklearn中导入三个监督学习模型  # DecisionTree  from sklearn import tree  #Guassian naive byes  from sklearn import naive\_bayes  #SVM  from sklearn import svm  # TODO：初始化三个模型  clf\_A = tree.DecisionTreeClassifier(random\_state=42)  clf\_B = naive\_bayes.GaussianNB()  clf\_C = svm.SVC()  # TODO：计算1%， 10%， 100%的训练数据分别对应多少点  samples\_1 = int(len(X\_train) \* 0.01)  samples\_10 = int(len(X\_train) \* 0.1)  samples\_100 = len(X\_train)  # 收集学习器的结果  results = {}  for clf in [clf\_A, clf\_B, clf\_C]:  clf\_name = clf.\_\_class\_\_.\_\_name\_\_  results[clf\_name] = {}  for i, samples in enumerate([samples\_1, samples\_10, samples\_100]):  results[clf\_name][i] = train\_predict(clf, samples, X\_train, y\_train, X\_val, y\_val)  print (results['GaussianNB'])  print (results['SVC'])  print (results['DecisionTreeClassifier'])  # 对选择的三个模型得到的评价结果进行可视化  vs.evaluate(results, accuracy, fscore) |



在训练时间上，决策树和高斯朴素贝叶斯消耗时间都较少。

在准确率和 F-Score得分也并没有比决策树表现更好，所以不考虑支持向量机模型。

从准确率上和F-Score上，决策树的性能比高斯贝叶斯更好，所以最终选择决策树。因此应该选择决策树模型进行预测。

决策树算法通过获取数据的不同属性，通过重复选择属性进行分类，得到能将当前种类分割的比例最好的属性作为当前这一次的分割条件，以此重复，最终获得一个每一层都能分类的树形结构。当有一个新的数据需要进行预测的时候，每一步传入对应的属性，最后得到的结果就是已经分好类的结果。

## 3.3 参数优化

调节选择的模型的参数。使用网格搜索（GridSearchCV）来至少调整模型的重要参数（至少调整一个），这个参数至少需尝试3个不同的值。你要使用整个训练集来完成这个过程。在接下来的代码单元中，你需要实现以下功能：

* 导入sklearn.model\_selection.GridSearchCV 和 sklearn.metrics.make\_scorer.
* 初始化你选择的分类器，并将其存储在clf中。

设置random\_state (如果有这个参数)。

* 创建一个对于这个模型你希望调整参数的字典。

例如: parameters = {'parameter' : [list of values]}。

注意： 如果你的学习器有 max\_features 参数，请不要调节它！

* 使用make\_scorer来创建一个fbeta\_score评分对象（设置 β=0.5）。
* 在分类器clf上用'scorer'作为评价函数运行网格搜索，并将结果存储在grid\_obj中。
* 用训练集（X\_train, y\_train）训练grid search object,并将结果存储在grid\_fit中。

|  |
| --- |
| # TODO：导入'GridSearchCV', 'make\_scorer'和其他一些需要的库  from sklearn.model\_selection import GridSearchCV,KFold  from sklearn.metrics import make\_scorer  # TODO：初始化分类器  clf = tree.DecisionTreeClassifier(random\_state=42)  # TODO：创建你希望调节的参数列表  # 控制树的最大深度(防止过拟合)和最少分类个数  parameters = {'max\_depth':range(5,10),'min\_samples\_split':range(2,10)}  # TODO：创建一个fbeta\_score打分对象  scorer = make\_scorer(fbeta\_score,beta=0.5)  # TODO：在分类器上使用网格搜索，使用'scorer'作为评价函数  kfold = KFold(n\_splits=10) #切割成十份  # 同时传入交叉验证函数  grid\_obj = GridSearchCV(clf,parameters,scorer,cv=10) #直接设置默认分层抽样 KFold  # TODO：用训练数据拟合网格搜索对象并找到最佳参数  grid\_obj.fit(X\_train,y\_train)  # 得到estimator  best\_clf = grid\_obj.best\_estimator\_  # 使用没有调优的模型做预测  predictions = (clf.fit(X\_train, y\_train)).predict(X\_val)  best\_predictions = best\_clf.predict(X\_val)  # 汇报调参前和调参后的分数  print ("\nUnoptimized model\n------")  print ("Accuracy score on validation data: {:.4f}".format(accuracy\_score(y\_val, predictions)))  print ("F-score on validation data: {:.4f}".format(fbeta\_score(y\_val, predictions, beta = 0.5)))  print ("\nOptimized Model\n------")  print ("Final accuracy score on the validation data: {:.4f}".format(accuracy\_score(y\_val, best\_predictions)))  print ("Final F-score on the validation data: {:.4f}".format(fbeta\_score(y\_val, best\_predictions, beta = 0.5))) |

|  |  |  |
| --- | --- | --- |
| 评价指标 | 未优化的模型 | 优化的模型 |
| 准确率 | 0.8208 | 0.8617 |
| F-score | 0.6386 | 0.7420 |

## 3.4 特征重要性与特征训练

在数据上（比如我们这里使用的人口普查的数据）使用监督学习算法的一个重要的任务是决定哪些特征能够提供最强的预测能力。专注于少量的有效特征和标签之间的关系，我们能够更加简单地理解这些现象，这在很多情况下都是十分有用的。在这个项目的情境下这表示我们希望选择一小部分特征，这些特征能够在预测被调查者是否年收入大于$50,000这个问题上有很强的预测能力。

### 3.4.1 提取特征的重要性

选择一个有 'feature\_importance\_' 属性的scikit学习分类器（例如 AdaBoost，随机森林）。'feature\_importance\_' 属性是对特征的重要性排序的函数。在下一个代码单元中用这个分类器拟合训练集数据并使用这个属性来决定人口普查数据中最重要的5个特征。

选择一个scikit-learn中有feature\_importance\_属性的监督学习分类器，这个属性是一个在做预测的时候根据所选择的算法来对特征重要性进行排序的功能。

在下面的代码单元中，你将要实现以下功能：

* 如果这个模型和你前面使用的三个模型不一样的话从sklearn中导入一个监督学习模型。
* 在整个训练集上训练一个监督学习模型。
* 使用模型中的 'feature\_importances\_'提取特征的重要性。

# TODO：导入一个有'feature\_importances\_'的监督学习模型

# TODO：在训练集上训练一个监督学习模型

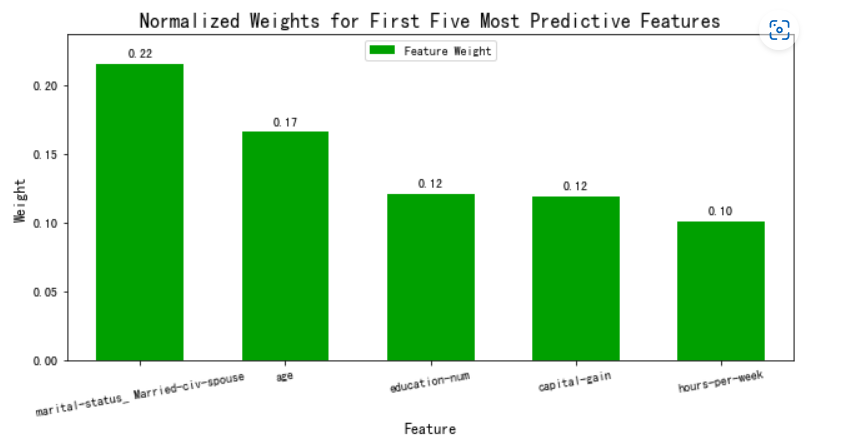
model = clf

# TODO： 提取特征重要性

importances = clf.feature\_importances\_

# 绘图

vs.feature\_plot(importances, X\_train, y\_train)



### 3.4.2 数据划分

如果我们只是用可用特征的一个子集的话模型表现会怎么样？通过使用更少的特征来训练，在评价指标的角度来看我们的期望是训练和预测的时间会更少。从上面的可视化来看，我们可以看到前五个最重要的特征贡献了数据中**所有**特征中超过一半的重要性。这提示我们可以尝试去**减小特征空间**，简化模型需要学习的信息。下面代码单元将使用你前面发现的优化模型，并**只使用五个最重要的特征**在相同的训练集上训练模型。

# 导入克隆模型的功能

from sklearn.base import clone

# 减小特征空间

X\_train\_reduced = X\_train[X\_train.columns.values[(np.argsort(importances)[::-1])[:5]]]

X\_val\_reduced = X\_val[X\_val.columns.values[(np.argsort(importances)[::-1])[:5]]]

# 在前面的网格搜索的基础上训练一个“最好的”模型

clf\_on\_reduced = (clone(best\_clf)).fit(X\_train\_reduced, y\_train)

# 做一个新的预测

reduced\_predictions = clf\_on\_reduced.predict(X\_val\_reduced)

# 对于每一个版本的数据汇报最终模型的分数

print ("Final Model trained on full data\n------")

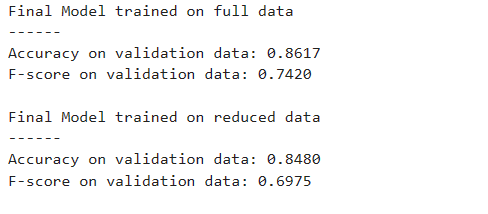
print ("Accuracy on validation data: {:.4f}".format(accuracy\_score(y\_val, best\_predictions)))

print ("F-score on validation data: {:.4f}".format(fbeta\_score(y\_val, best\_predictions, beta = 0.5)))

print ("\nFinal Model trained on reduced data\n------")

print ("Accuracy on validation data: {:.4f}".format(accuracy\_score(y\_val, reduced\_predictions)))

print ("F-score on validation data: {:.4f}".format(fbeta\_score(y\_val, reduced\_predictions, beta = 0.5)))



**3.5 模型测试**

#TODO test your model on testing data and report accuracy and F score

copy\_test = [value for value in X\_test]

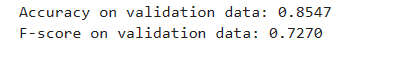
copy\_predicts = best\_clf.predict(X\_test)

print ("Accuracy on validation data: {:.4f}".format(accuracy\_score(y\_test, copy\_predicts)))

print ("F-score on validation data: {:.4f}".format(fbeta\_score(y\_test, copy\_predicts, beta = 0.5)))

# 选择这个模型是因为在时间上比较效率,在准确率和 F-Score 上也都能获得相对较高的得分.

# 最终结果准确率为0.8551,F-Score 为0.7278,都比较良好,能进行相对较好的预测.



# 总结与展望

本次项目中，我负责数据下载和一部分的代码实现，并且完成PPT的初始制作。

在这个项目中，我们使用1994年美国人口普查收集的数据构建了一个分类器来预测被调查者的收入。我们采用支持向量机、决策树、朴素贝叶斯等监督学习算法，并通过交叉验证来优化每个算法的超参数。最终，我们选择了支持向量机作为最佳候选算法，得到了一个准确率为85.5%的分类器。

该项目所取得的成果表明，机器学习算法可以成功地应用于社会经济领域的问题，可帮助政府、研究人员和企业等各方更好地了解和解决社会问题。

局限性：

任何机器学习系统都有其局限性。在这个项目中，我们只使用了1994年的数据，这可能不足以反映当前的社会经济状况。此外，我们使用了一些经典的监督学习算法，在处理大规模、高维度的数据时可能存在局限性。未来的工作可能需要探索更先进的算法和技术来解决这些问题。

未来工作：

我们计划将此项目扩展到其他领域，并使用更多的数据集来训练和测试分类器的性能。此外，我们将探索如何加入深度学习技术来进一步提高分类器的性能。我们也计划扩展该分类器，包括预测其他变量，如婚姻状况、种族、教育程度等。

机器学习的重要性：

在这个项目中，我们证明了机器学习的潜力，可用于解决社会经济领域的问题。未来，随着大数据和人工智能技术的不断发展，机器学习将成为更多问题的解决方案。从商业应用到医疗保健，机器学习可以帮助我们更好地了解数据并做出更准确的预测，以改善我们的生活。

进一步研究：

最后，我们建议进一步研究机器学习的技术和算法，并考虑如何将其应用到更广泛的问题领域。具体而言，我们需要更深入地研究监督学习和无监督学习算法、神经网络、深度学习等先进技术，并探索它们如何应用于其他领域，如自然语言处理、图像识别、智能交通等。