Predicting the reading habit based on socioeconomic features using classification methods Logistic Regression and Decision Trees

1. Introduction

Books are the ladder of human progress. Nowadays we can find news and information on social media easily, but how about the traditional learning and entertainment method? The reading habit is affected by many factors and predicting it accurately is not straightforward. The goal of this project is to try different classification methods to predict whether a person is currently an active reader based on their socioeconomic features. This kind of problem requires experts to determine and has no existing machine learning models, but is quite essential when the government wants to know more about the public interest to guide better, or businesses need to analyze the market and consumers.

This report will briefly go through the project. Section 2 presents how this problem is formulated and explains data points, features and label. Section 3 discusses the dataset's details and two methods used, namely Logistic Regression and Decision Trees. Section 4 then shows and compares the errors performed by models. Section 5 draws a conclusion and makes suggestions for future improvements.

2. Problem Formulation

According to the textbook, a machine learning problem consists of three parts: data, model and loss[3]. Here we discuss the data part and present how this problem is formulated. The application can be modeled with **datapoints** representing individuals, or more specifically, the description of the socioeconomic status and reading habits. Each data point is characterized by **features** including age **as an integer**, gender, marital status, education, employment and incomes transformed **as numerical categories**. A **binary category label** is defined as whether the person is currently an active reader or not (No: 0, Yes: 1), determined by other four properties: the number of books read and whether the person has read printed books, audiobooks, or e-books during last 12 months as follows.

3. Methods

Dataset

The dataset used in this project can be downloaded from https://www.kaggle.com/vipulgote4/reading-habit-dataset. First I did some preprocessing work from cleaning the data. The original dataset contains 2831 interviewees' information with 14 attributes of background and reading habits. There are no missing features as we described in the problem formulation in the dataset.

	Age	Sex	Race	Marital status?	Education	Employement	Incomes	How many books did you read during last 12months?	Read any printed books during last 12months?	Read any audiobooks during last 12months?	Read any e-books during last 12months?	Last book you read, you...	Do you happen to read any daily news or newspapers?
0	66	Male	Refused	Divorced	College graduate	Retired	20,000under 30,000	97	Yes	No	Yes	Purchased the book	No
1	46	Male	Native American/American Indian	Married	High school graduate	Employed full-time	Less than $10,000	97	Yes	Yes	Yes	Purchased the book	Yes
2	32	Male	Mixed race	Never been married	High school graduate	Employed full-time	Less than $10,000	97	No	Yes	Yes	Borrowed the book from a friend or family member	Yes

Figure 1. Raw data

But some of the attributes such as 'Race','Last book you read, you...','Do you happen to read any daily news or newspapers?','Do you happen to read any magazines or journals?' are not relevant for this problem, so these columns were droped. There were 390 missing points detected where Read

any printed books, audiobooks, e-books are NaN, because they are individuals whose book count is 0. So instead of dropping these rows, I filled those blanks with 'No'. Some answers to income or education are 'Don't know' or 'Refused'. I deleted these rows for no contribution to the model.

Then for convenience and feasibility of the latter classification model, some categorical variables were transformed into numerical. The six **features** are:

1) age: Numerical
2) sex: 'Male' to 0, 'Female' to 1
3) marital: Others to 0, 'Married' to 1
4) education: High school grad or less to 0, Some college to 1, College+ to 2
5) employment: 'Employed full-time' to 0, 'Student' to 1, 'self-employed' to 2, 'Employed part-time' to 3, 'Not employed for pay' to 4, 'Disabled' to 5, 'Retired' to 6.
6) incomes: Below $30,000 to 0, $30,000-$49,999 to 1, $50,000-$74,999 to 2, $75,000+ to 3. Initially not a numerical variable, recategorized referring to the research report[2].

The reason why I chose these characteristics is a combination of common-sense, research findings and data visualization. Intuitively, persons with more money, spare time and knowledge (education, employment, incomes) are more likely to read books. Ages, sex and marital status are correlated to reading habits. For example, the young likes e-books, while the seniors would prefer printed books. Meanwhile, this correlation was shown in the survey conducted by the Pew Research Center[2], which is also the original source of this dataset.

I created a new property to be considered as the **label**, named as isreader, which represents whether this person is currently an active reader or not. The label is determined by the value in the column 'booknum', 'ifprinted', 'ifaudio', 'ife'. In particular, we define the label of a datapoint to be equal to 1 if the corresponding person has read over twelve books, and has reading experience of printed books, audiobooks as well as e-books during the last twelve months, otherwise we define the label to be 0. More formally, y=1 if booknum>12 and ifprinted&ifaudio&ife==1, and y=0 if booknum<=12 or ifprinted|ifaudio|ife==0. This label was chosen because it is interesting to explore a new variable, for another thing, it is a reasonable overall measurement of one's reading habit. An active reader is supposed to read a certain number of books and enjoy different types of mediums.

	age	sex	marital	education	employment	incomes	booknum	ifprinted	ifaudio	ife	isreader
0	66	0	0	2	6	0	97	1	0	1	0
1	46	0	1	0	0	0	97	1	1	1	1
2	32	0	0	0	0	0	97	0	1	1	0

Figure 2. Dataset after preprocessing

After the preprocessing, we finally got 2457 datapoints in total with no missing data in any fields. Then let us have a general looking at the data visualized by matplotlib.pyplot and seaborn package. First, we examined the label variable. Then we had a look at the numerical feature.

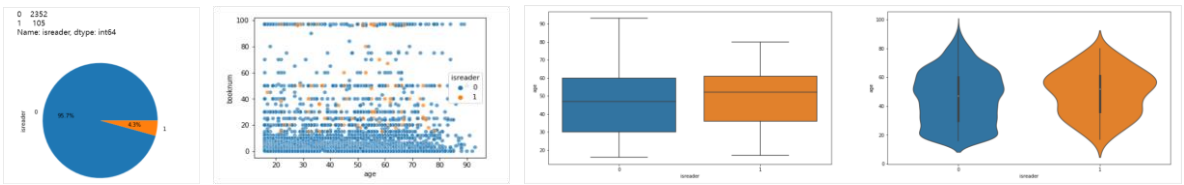


Figure 3. Data visualization 1

We can see the distribution of the five categoricals. And the following selected part of a diagram showing the correlation between features, where the yellow dots refer to the positive 'isreader'.

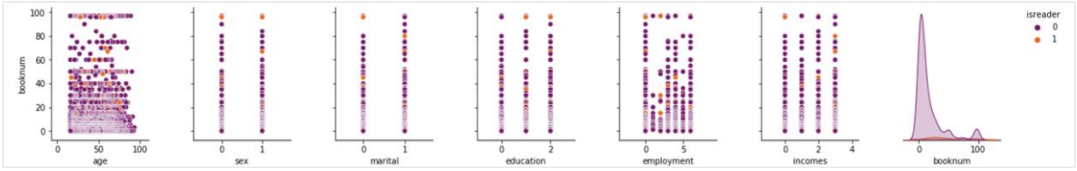


Figure 4. Data visualization 2

Models

In this project, predictions would be made with two different methods trained on the same dataset. Since datapoints have multi-class features and binary labels, the best-suited models are logistic regression (ch. 3.6) and decision tree (chapter 3.10 of mlbook.cs.aalto.fi).

Logistic regression

Logistic regression is a ML method that allows classifying datapoints according to two categories applied to datapoints characterized by feature vectors and binary labels[3]. It was employed **because** it is suitable for binary prediction problems with multi-features just like this project, and is fairly simple and easy to implement. Python's scikit-learn library has multiple useful functions. The class LogisticRegression was used to perform this method, which can be seen from the code attached.

Decision Tree

The second method I applied to the data was decision tree classifier. A decision tree is made out of nodes connected by edges, creating a discontinuous stepwise function to predict the labels. The computation starts at a root node and continues towards the leaf nodes of the tree. The **reason** why I used Decision Tree method is that it works as an efficient classifier for this type of data and it leaves out the unimportant features easily. The DecisionTreeClassifier class was used to perform the model. But besides, it tends to overfit as the number of splits increases. **To avoid overfitting**, an optimal stopping depth should be determined. We can build a hypothesis space by considering all decision trees not exceeding a maximum depth. Different maximum tree depths from 1 to 20 were tested. The one with the smallest validation error was chosen.

Loss function

To evaluate the models, **Logistic loss** was chosen as the loss function for both models. Because It was the default choice for the first method logistic regression, and fits well with the decision tree model as a popular loss function for binary classification problems to search for an accurate hypothesis [3]. I used the function log_loss in scikit-learn for both training and validation errors, which is defined as

$$L\log(y, p) = -(y \log(p) + (1 - y) \log(1 - p))$$

for a sample with a true label y = 0,1. And the **accuracy score metric** from sklearn.metrics package was also calculated for general evaluation and comparison.

Model validation

The data was randomly shuffled and split into three sets: training (64%), testing (20%), and validation (16%) sets with function train_test_split. I used this ratio because first it followed the principle that the training set should be larger than the validation set. And it is easy to carry out and commonly used in ML problems. The model was trained with the train set and the performance was evaluated with the validation set. After the better model was chosen, it would be assessed with the test set.

4. Results

The training and validation errors for two models can be seen in the table below.

	Training error	Validation error	Validation accuracy
Logistic Regression	0.1607	0.1949	0.9491
Decision Tree	0.1576	0.1439	0.9644

Table 1. Model performance

For the decision tree part, after a simple experiment, we found that d=3 was the most appropriate depth. Because after the depth is 4 or bigger, while the training error is still decreasing, the validation error starts to increase significantly. So I chose d=3 due to its smallest validation error. Even though the validation error is slightly smaller, they are still close, so it does not seem to underfit badly. For logistic regression, the validation error is slightly larger, suggesting the model might be overfitting.

To choose between these two models, errors and accuracy were compared. The differences were not great, but from the table we can see the second line has smaller traning and validation error, and higher accuracy score. Based on the results, I chose **Decision Tree** with depth=3 as the final model, since it predicted the problem better with lower validation error and better accuracy. Finally, the chosen model was evaluated on the testing set. It has been introduced in the Method section how the test set was constructed, split from the dataset (20%) separately and used neither in training set nor in validation set. The test error was 0.1423, and the accuracy score was 0.9634.

5. Conclusions

In general, we realized the goal of this project by comparing different classification methods to predict whether a person is currently an active reader based on their socioeconomic features. According to the results, both of the models seem to predict the labels somewhat well. The finally chosen model had an accuracy of 0.94 and test error of 0.21, which indicates that we can guarantee 94% possibility to make the right prediction. This is sufficient in the application domain, unlike in the disease diagnosis, but there are many limitations and improvements could also be tried in the future.

To begin with, the validation error was slightly smaller than the training error which hints at underfitting. To improve the model I would try k-fold cross validation instead of a simple split, to find better parameters for the function. Second, the dataset was imbalanced. To represent the minority (isreader) group properly and to make sure the result is convincing, some methods could be used, for instance, SMOTE [5]. Furthermore, just as other ml problems, to make the predictions better, I would like to add more data points. As a result, the model could be trained better and more accurately. Additionally, it is interesting to explore people's reading habits from different countries.

References

[1] kaggle.com/vipulgote4/reading-habit-dataset

[2] https://www.pewresearch.org/fact-tank/2015/10/19/slightly-fewer-americans-are-reading-print-books-new-survey-finds/

[3] Alex Jung. *Machine Learning. The Basics.* mlbook.cs.aalto.fi

[4] Scikit Learn, https://scikit-learn.org/stable/index.html

[5] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16:321–357, Jun 2002.

Appendix

In [11]:
```
%config Completer.use_jedi = False  # enable code auto-completion
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns  #data visualization library
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix  # evaluation metrics
```

Dataset

Reading the data

In [12]:
```
# Read in the data stored in the file 'ReadingHabbitData.csv'
df = pd.read_csv('ReadingHabbitData.csv')
# Print the first 5 rows of the DataFrame 'df'
df.head(5)
```

Out[12]:

	Age	Sex	Race	Marital status?	Education	Employement	Incomes	How many books did you read during last 12months?	Read any printed books during last 12months?
0	66	Male	Refused	Divorced	College graduate	Retired	$20,000tounder30,000$	97	Yes
1	46	Male	Native American/American Indian	Married	High school graduate	Employed full-time	Less than $10,000	97	Yes
2	32	Male	Mixed race	Never been married	High school graduate	Employed full-time	Less than $10,000	97	No
3	27	Male	Mixed race	Married	High school graduate	Employed full-time	$40,000tounder50,000$	97	Yes
4	16	Female	Mixed race	Never been married	High school incomplete	Employed part-time	$10,000tounder20,000$	97	Yes

Preprocessing

In [ ]:
```
# drop unrelevant columns
df.drop(columns=['Race','Last book you read, you...','Do you happen to read any daily news or newspapers?',\
                 'Do you happen to read any magazines or journals?'],inplace=True)
# rename columns
df.columns =['age','sex','marital','education','employment','incomes','booknum','ifprinted','ifaudio','ife']
df.head(5)
```

In [14]:
```
# get the number of missing data points per column
missing_number = df.isnull().sum()
missing_percent = (df.isnull().sum()/df.isnull().count())
missing_values = pd.concat([missing_number, missing_percent], axis=1, keys=['Missing_Number', 'Missing_Percent'])
missing_values
```

Out[14]:

	Missing_Number	Missing_Percent
Age	0	0.000000
Sex	0	0.000000
Marital status?	0	0.000000
Education	0	0.000000
Employement	0	0.000000
Incomes	0	0.000000
How many books did you read during last 12months?	0	0.000000
Read any printed books during last 12months?	390	0.137712
Read any audiobooks during last 12months?	390	0.137712
Read any e-books during last 12months?	390	0.137712

In [ ]:
```
# fill the missing blank
df.fillna('No', inplace=True)
df.tail()

# delete the rows containing invalid value
df=df[~(df['marital'].str.contains('know')|df['education'].str.contains('know|None')|df['incomes'].str.contains('Refused')\
        |df['ifprinted'].str.contains('know')|df['ifaudio'].str.contains('know')|df['ife'].str.contains('know'))]
df.tail()

# transform categories to numerical
df['sex'] = df['sex'].map({'Male': 0, 'Female': 1}).astype(int)
df['marital'] = df['marital'].map( {'Single': 0, 'Separated': 0, 'Never been married': 0, 'Divorced': 0, 'Widowed': 0,'Living with a partner': 0, 'Married': 1} ).astype(int)
df['education'] = df['education'].map( {'High school incomplete': 0, 'High school graduate': 0, 'Some college, no 4-year degree': 1, 'Technical, trade or vocational school AFTER high school': 1, 'College graduate': 2, 'Post-graduate training/professional school after college': 2} ).astype(int)
df['employment'] = df['employment'].map( {'Employed full-time': 0, 'Student': 1, 'Have own business/self-employed': 2, 'Employed part-time': 3, 'Other': 4, 'Not employed for pay': 4, 'Disabled': 5, 'Retired': 6} ).astype(int)
df['incomes'] = df['incomes'].map( {'Less than $10,000': 0, '$10,000 to under $20,000': 0, '$20,000 to under $30,000': 0, \
                                   '$30,000 to under $40,000':1, '$40,000 to under $50,000': 1,\
                                   '$50,000 to under $75,000': 2, \
                                   '$75,000 to under $100,000': 3, '$100,000 to under $150,000': 3, '9$100,000 to under $150,000': 3} ).astype(int)
df['ifprinted'] = df['ifprinted'].map({'No': 0, 'Yes': 1}).astype(int)
df['ifaudio'] = df['ifaudio'].map({'No': 0, 'Yes': 1}).astype(int)
df['ife'] = df['ife'].map({'No': 0, 'Yes': 1}).astype(int)
df
```

In [49]:
```
# define the label, a new column
minvalue = df['booknum'].min() # minimum value of the column 'booknum'
maxvalue = df['booknum'].max() # maximum value of the column 'booknum'

if 'isreader' in df.columns:  # delete existing 'binarized max temperature' if there is
    df = df.drop(['isreader'],axis=1)

bi_labels = [0,1] # new labels to be assigned
bi_cut_bins = [minvalue,12,maxvalue] #cutting intervals/criteria [minvalue,12],(12,maxvalue]

# Encode max temperatures to binary labels
binarized_booknum = pd.cut(df['booknum'],bins=bi_cut_bins,labels=bi_labels,include_lowest=True)
# Insert a new column "isreader" to the datafame
# Int 10 is the position where the new column will be inserted to
df.insert(10,'isreader',binarized_booknum)
df['isreader'] = np.where(df['isreader']&df['ifprinted']&df['ifaudio']&df['ife'], 1, 0)
df.head()
```

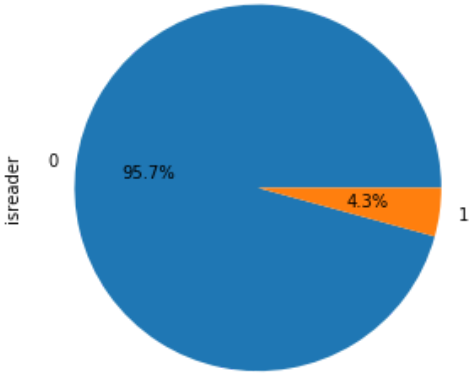
Out[49]:

	age	sex	marital	education	employment	incomes	booknum	ifprinted	ifaudio	ife	isreader
0	66	0	0	2	6	0	97	1	0	1	0
1	46	0	1	0	0	0	97	1	1	1	1
2	32	0	0	0	0	0	97	0	1	1	0
3	27	0	1	0	0	1	97	1	0	1	0
4	16	1	0	0	3	0	97	1	1	0	0

Data Visualization and Analysis

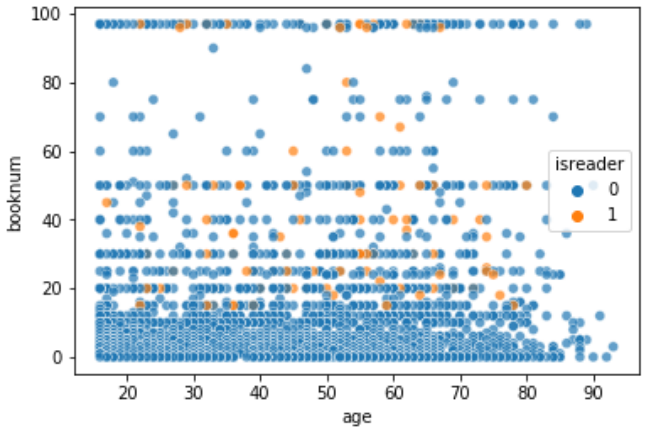
In [25]:
```
# plot the label
print(df["isreader"].value_counts())
df["isreader"].value_counts().plot(kind="pie", autopct='%1.1f%%', figsize=(5,5));
```

```
0    2352
1     105
Name: isreader, dtype: int64
```

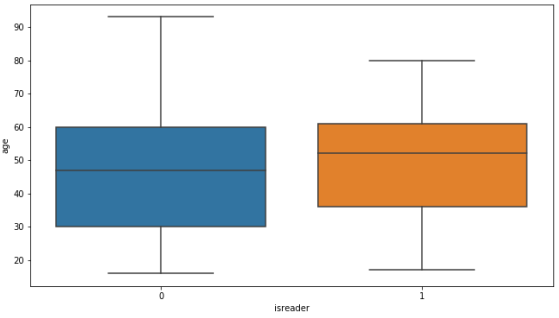


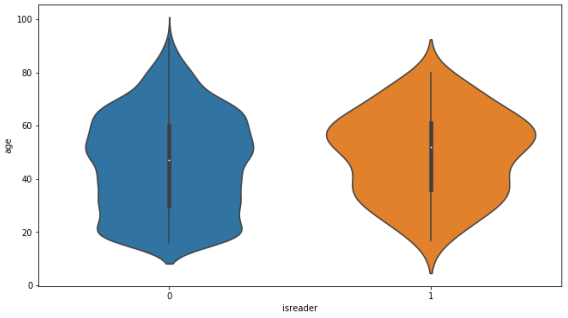
In [28]:
```
# plot the numerical feature
# scatter
sns.scatterplot(data=df, x='age', y='booknum', hue='isreader', alpha=0.7)
```

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x150cf9d7a58>

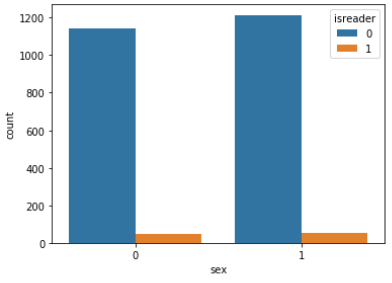


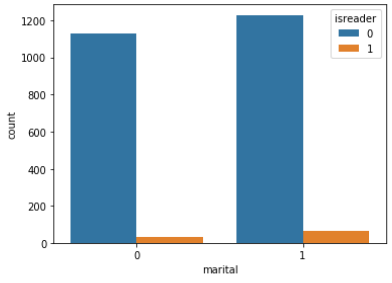
In [33]:
```
fig, ax = plt.subplots(1, 2, figsize=(24, 6))
# boxplot
sns.boxplot(x='isreader', y='age', data=df, ax=ax[0])
# violinplot
sns.violinplot(x='isreader', y='age', data=df, ax=ax[1])
plt.show()
```

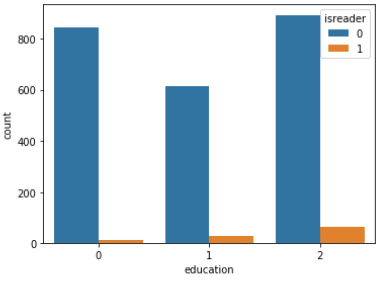


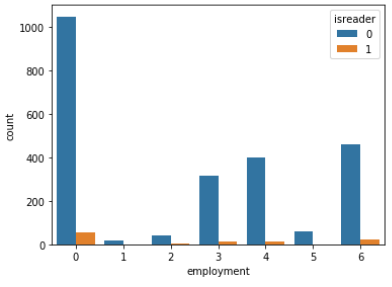


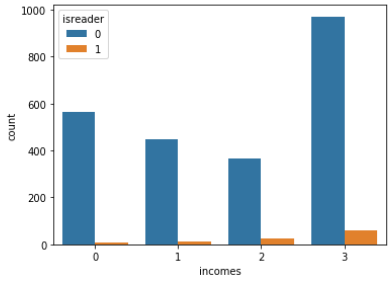
In [39]:
```
# plot the categorical feature
categorical = ['sex', 'marital', 'education', 'employment', 'incomes']
index = 0
plt.figure(figsize=(20,20))
for feature in categorical:
    if feature != "isreader":
        index += 1
        plt.subplot(4, 3, index)
        sns.countplot(data = df, x =feature, hue = "isreader")
```



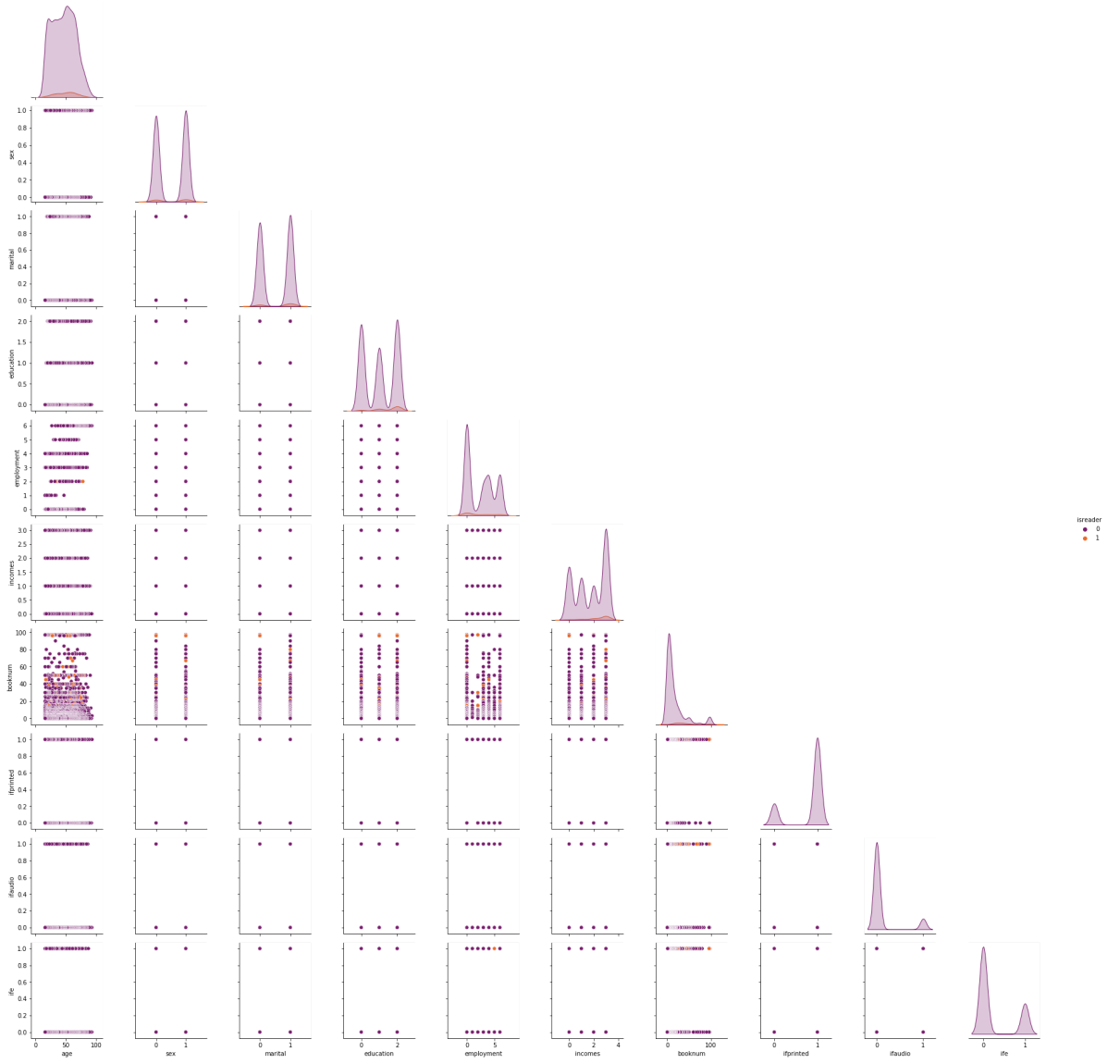








In [38]:
```
# pairplot
sns.pairplot(df, hue="isreader", palette="inferno", corner=True);
```



In [ ]:

Models

Logistic regression

In [149]:

```
X = df[['age','sex','marital','education','employment','incomes']]
y = df['isreader']

temp_train_feas, X_test_i, temp_train_labs, y_test = train_test_split(X.index, y, test_size = 0.2)
X_train_i, X_val_i, y_train, y_val = train_test_split(temp_train_feas, temp_train_labs, test_size = 0.2)

X_train = X.loc[X_train_i]
X_val = X.loc[X_val_i]
X_test = X.loc[X_test_i]

# print(y_train)
# print(y_val)
# print(y_test)


from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, f1_score, recall_score
from sklearn.metrics import log_loss

clf = LogisticRegression()
clf.fit(X_train, y_train)

y_pred = clf.predict(X_val)
acc_lr = accuracy_score(y_val, y_pred)

y_pred_prob = clf.predict_proba(X_val)
val_loss_lr = log_loss(y_val, y_pred_prob)

y_pred_prob = clf.predict_proba(X_train)
train_loss_lr = log_loss(y_train, y_pred_prob)

print("Accuracy score: ", acc_lr)
print("Validation loss:  ", val_loss_lr)
print("Training loss:     ", train_loss_lr)
```

```
Accuracy score:  0.9491094147582697
Validation loss:   0.19492309121923687
Training loss:     0.16074411712243253
```

Decision tree

In [109]:
```
X = df[['age','sex','marital','education','employment','incomes']]
y = df['isreader']

temp_train_feas, X_test_i, temp_train_labs, y_test = train_test_split(X.index, y, test_size = 0.2)
X_train_i, X_val_i, y_train, y_val = train_test_split(temp_train_feas, temp_train_labs, test_size = 0.2)

X_train = X.loc[X_train_i]
X_val = X.loc[X_val_i]
X_test = X.loc[X_test_i]

from sklearn import tree
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import log_loss

bestloss = 500
bestdepth = 0
bestacc = 500
bestf1 = 500

for i in range(1,20):
#     print("Tree depth = ", i)
    clf_tree = tree.DecisionTreeClassifier(max_depth = i)
    clf_tree.fit(X_train, y_train)

    y_pred_tree = clf_tree.predict(X_val)
    acc_tree = accuracy_score(y_val, y_pred_tree)

    y_pred_prob_t = clf_tree.predict_proba(X_val)
    val_loss_tree = log_loss(y_val, y_pred_prob_t)

    y_pred_prob_t = clf_tree.predict_proba(X_train)
    train_loss_tree = log_loss(y_train, y_pred_prob_t)

#     print("The accuracy: ", acc_tree)
#     print("The validation loss: ", val_loss_tree)
#     print("The training loss: ", train_loss_tree)

    if val_loss_tree < bestloss:
        bestdepth = i
        bestloss = val_loss_tree
        besttrainloss = train_loss_tree
        bestacc = acc_tree
        clf = clf_tree

print("Best depth: ", bestdepth)
print("Accuracy score: ", bestacc)
print("Validation error: ", bestloss)
print("Training error:    ", besttrainloss)
```

```
Best depth:  3
Accuracy score:  0.9643765903307888
Validation error:  0.14390378486877387
Training error:    0.15763455549716152
```

Test

In [144]:
```
# test the decision tree model
y_test_pred = clf.predict(X_test)
print("The test accuracy: ", accuracy_score(y_test, y_test_pred))

y_test_pred_prob = clf.predict_proba(X_test)
test_loss_tree = log_loss(y_test, y_test_pred_prob)
print("The testing loss:   ", test_loss_tree)
```

```
The test accuracy:  0.9634146341463414
The testing loss:   0.14232168210348847
```

In [ ]:
```

```