Capstone Project José Estevan

Mar 08st, 2023

Data Science Nanodegree

Starbucks Capstone Project

Project Definition

Project Overview

This data set contains simulated data that mimics customer behavior on the Starbucksrewards mobile app. Once every few days, Starbucks sends out an offer to users of the mobile app. An offer can be merely an advertisement for a drink or an actual offer such as a discount or BOGO (buy one get one free). Some users might not receive any offer during certain weeks. Not all users receive the same offer, and that was the challenge to solve with this data set. Our task is to combine transaction, demographic and offer data to determine which demographic groups respond best to which offer type. This data set is a simplified version of the real Starbucks app because the underlying simulator only has one product whereas Starbucks actually sells dozens of products. Every offer has a validity period before the offer expires. As an example, a BOGO offer might be valid for only 5 days. Informational offers also have a validity period even though these ads are merely providing information about a product; for example, if an informational offer has 7 days of validity, we can assume the customer is feeling the influence of the offer for 7 days after receiving the advertisement.

Data Dictionary

profile.json Rewards program users (17000 users x 5 fields)

- gender: (categorical) M, F, O, or null
- age: (numeric) missing value encoded as 118
- id: (string/hash) portfolio.json Offers sent during 30-day test period (10 offers x 6 fields)
- reward: (numeric) money awarded for the amount spent
- channels: (list) web, email, mobile, social
- difficulty: (numeric) money required to be spent to receive reward
- duration: (numeric) time for offer to be open, in days
- offer_type: (string) bogo, discount, informational
- id: (string/hash) transcript.json Event log (306648 events x 4 fields)
- person: (string/hash)

- event: (string) offer received, offer viewed, transaction, offer completed
- value: (dictionary) different values depending on event type
- offer id: (string/hash) not associated with any "transaction"
- amount: (numeric) money spent in "transaction"
- reward: (numeric) money gained from "offer completed"
- time: (numeric) hours after start of test

Offer Types

There are three types of offers that can be sent: buy-one-get-one (BOGO), discount, and informational.

- In a BOGO offer, a user needs to spend a certain amount to get a reward equal to that threshold amount.
- In a discount, a user gains a reward equal to a fraction of the amount spent.
- In an informational offer, there is no reward, but neither is there a requisite amount that the user is expected to spend. Offers can be delivered via multiple channels.

Problem Statement

Forecasting the purchase offer to which a possible greater level of reaction or user activities such as "offer received," "offer seen," "transaction," and "offer completed" can be achieved based on the demographic parameters of the consumer and other qualities of the companies' buy offers..

Metrics:

Accuracy is the quintessential classification metric. ... And easily suited for binary as well as a multiclass classification problem. Accuracy = (TP+TN)/(TP+FP+FN+TN) Accuracy is the proportion of true results among the total number of cases examined.

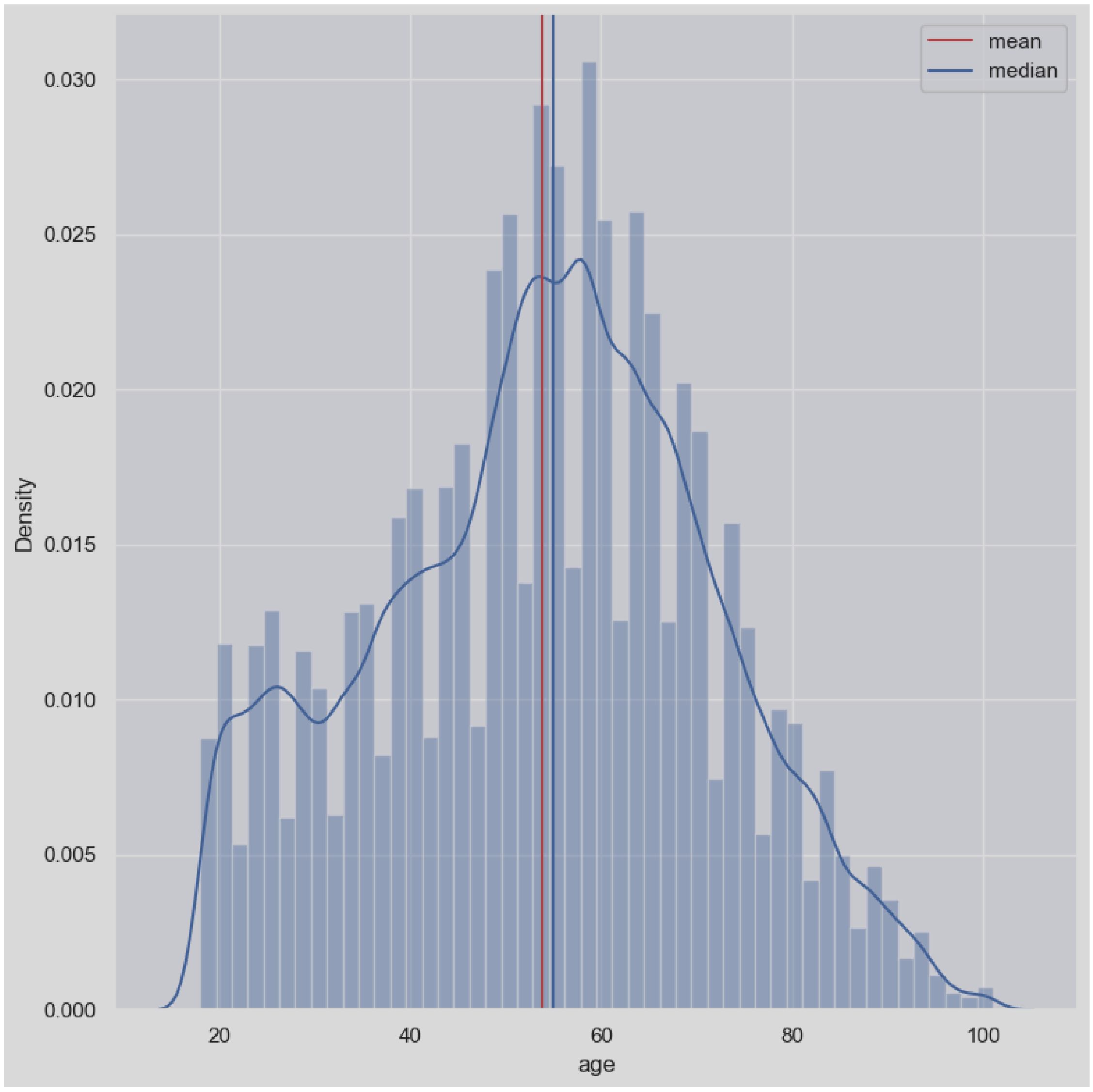
- became_member_on: (date) format YYYYMMDD
- income: (numeric)

Data Exploration and Visualization

1. Age Group

- Average Aged user is middle age ie. around 50-62 years

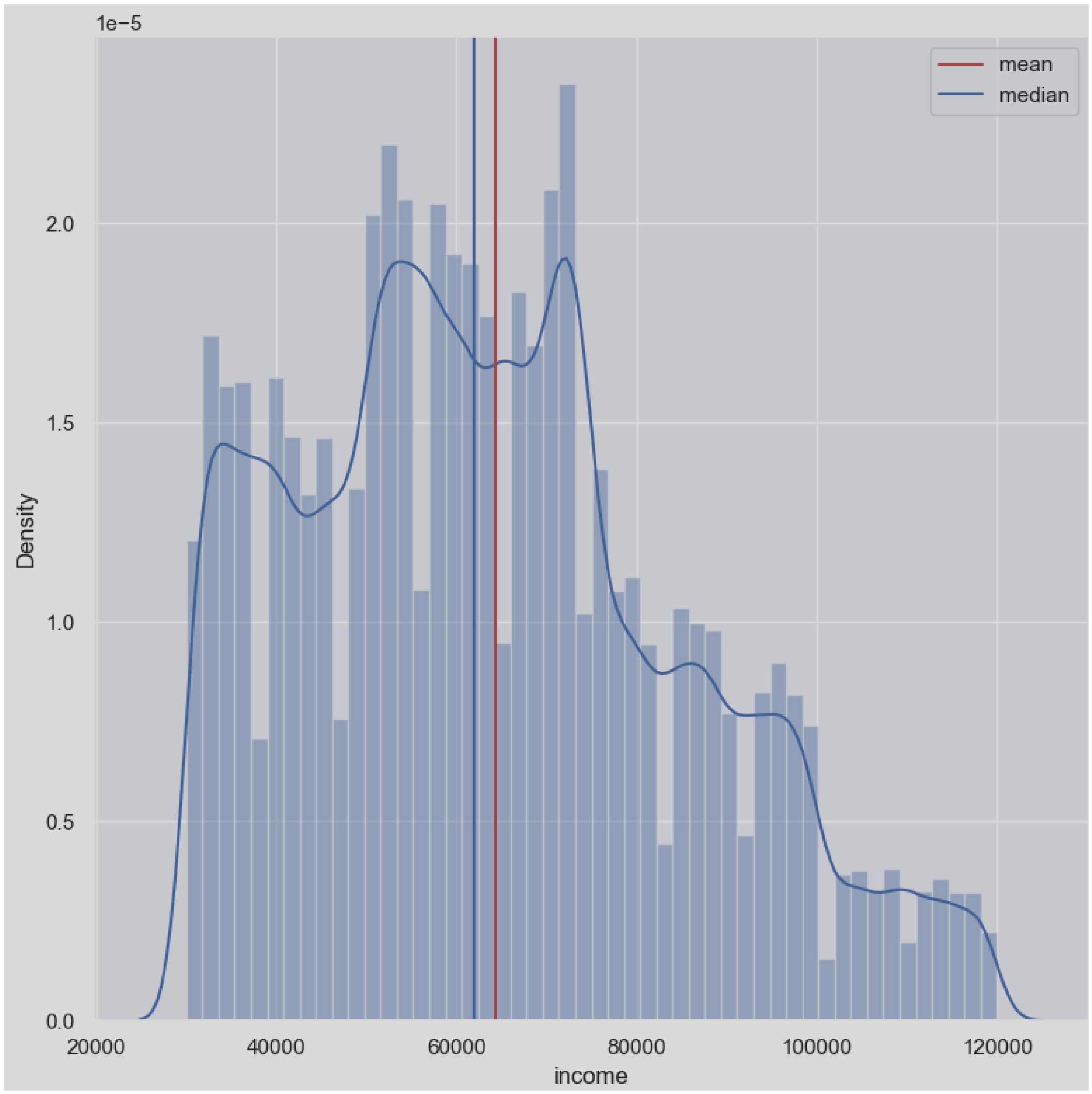
```
from IPython.display import Image
Image("screenshots/Screenshot_2.png")
```



2 . Income Range

- Average income user is middle income group ie. 65000-70000

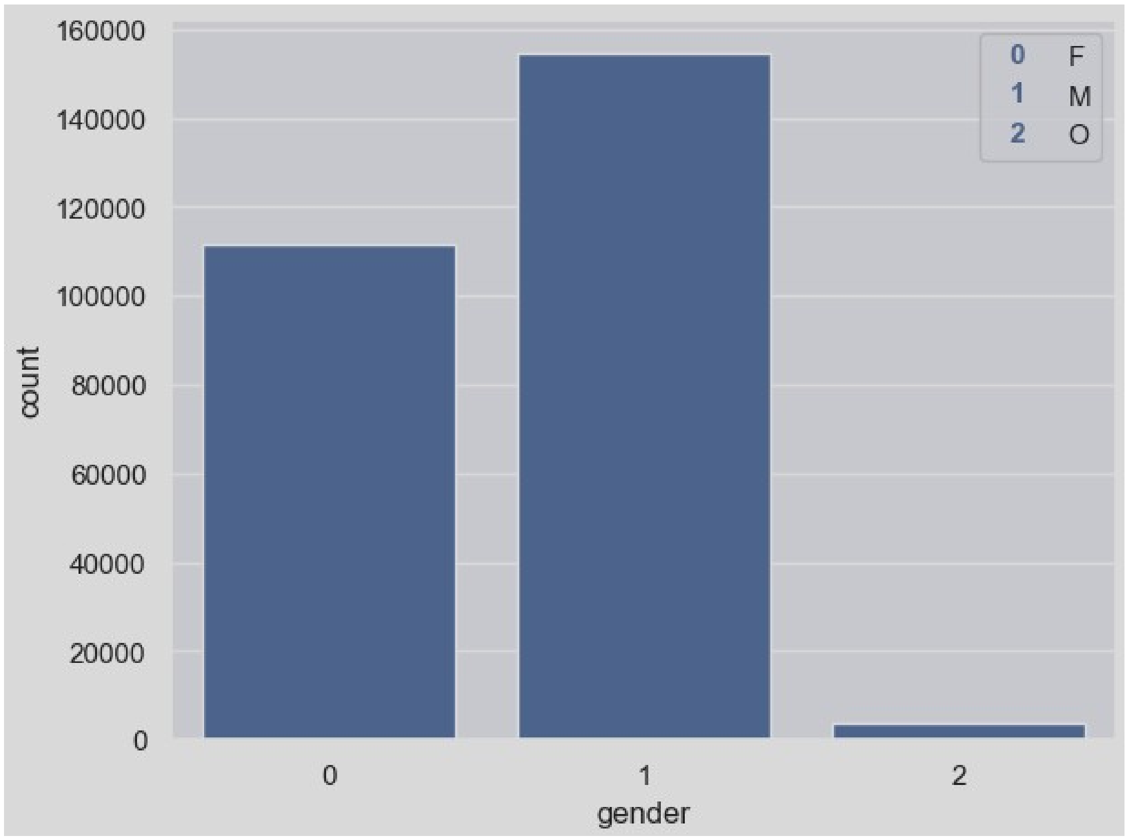
```
from IPython.display import Image
Image("screenshots/Screenshot_3.png")
```



3. Gender :

- Males are more than 50 percent of users
- male_proportion = 50.79%,
- female_proportion = 36.89%,
- others_proportion = 1.29%

```
from IPython.display import Image
Image("screenshots/Screenshot_4.png")
```



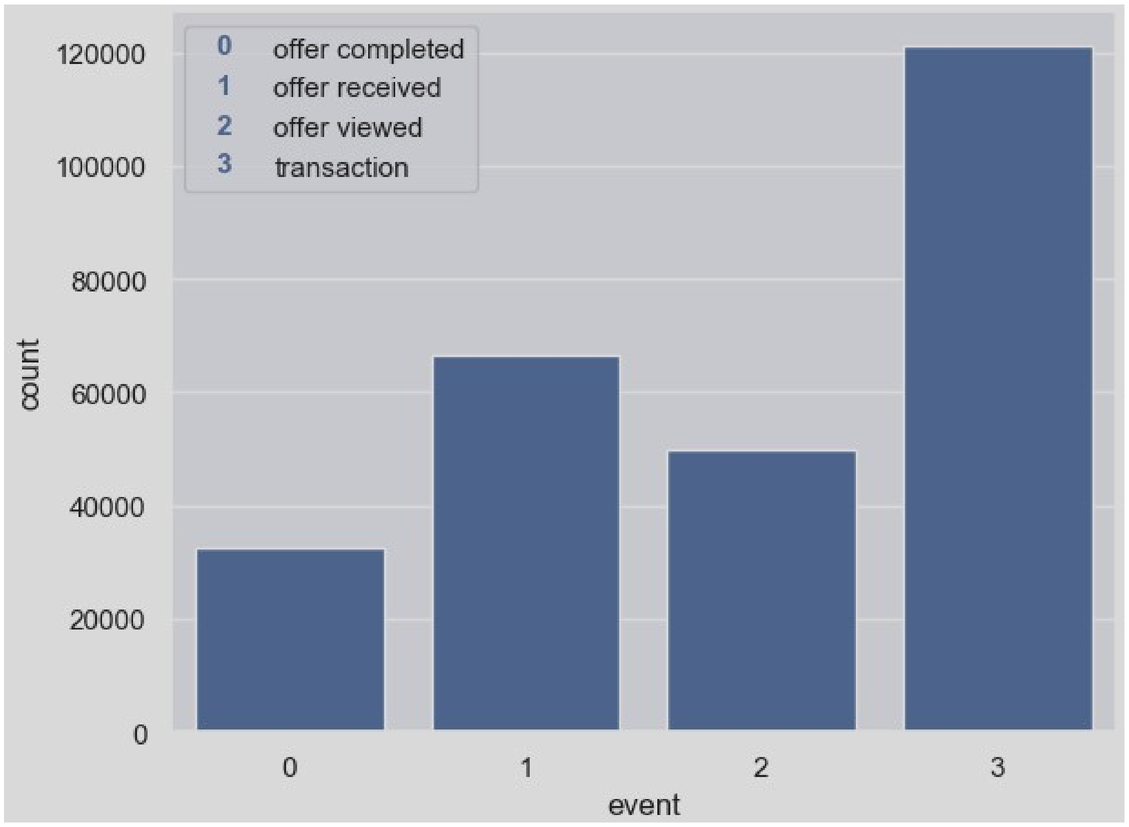
4. No. of Different Type of Offers Received by the Users

- BOGO offers are highly demanding , 30499 users received BOGO offers, 25449 viewed the offer and 15669 completed it.
- The percentage of BOGO Offer viewers is 83 percent .
- The percentage of DISCOUNT Offer viewers is 70 percent .

For BOGO Offer

- Offer _Viewed_proportion = 83.44% ,
- Offer_Completed_proportion = 51.37%

```
from IPython.display import Image
Image("screenshots/Screenshot_5.png")
```



Algorithms and Techniques

A random forest is a meta estimator that employs averaging to increase the predicted accuracy and control over-fitting. It does this by fitting a number of decision tree classifiers on various subsamples of the dataset. If the bootstrap=True flag is set, the max samples parameter is utilized to determine the size of each sub-sample; if it is false, the entire dataset is applied to the construction of each tree.

Inputs

Source data fed into the classifier, with the goal of making a decision or prediction about the data.

Training Set

A set of inputs for which the correct outputs are known, used to train the classifier.

Hyperparameters

A hyperparameter is a setting that affects the structure or operation of classifier. In real machine learning projects, tuning hyperparameters is the primary way to build an algorithm that provides accurate predictions for a certain problem.

- n_estimators: the number of decision trees in the forest. Increasing this hyperparameter generally improves the performance of the model but also increases the computational cost of training and predicting.
- max_depth: the maximum depth of each decision tree in the forest. Setting a higher value for max_depth can lead to overfitting while setting it too low can lead to underfitting.

Methodology

Data Preprocessing :

Splitting Dataset :

The train_test_split function is for splitting a single dataset for two different purposes: training and testing. The testing subset is for building your model. The testing subset is for using the model on unknown data to evaluate the performance of the model.

Feature Scaling :

Standardization & Normalization

Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.

Implementation :

Build a Model :

The implementation process can be split into two main stages:

1. The classifier training stage
2. The application development stage During the first stage, the classifier was trained on the preprocessed training data.
3. Define the algorithm and training parameters
4. Define the accuracy
5. Train the classifier, logging the validation accuracy
6. If the accuracy is not high enough, return to step 3

Refinement :

Improving Prediction Model :

We have a overfitting problem, it happens whena model is unable to generalize and instead fits the training dataset in an excessively close manner, this is an example of overfitting. Overfitting can occur for a variety of causes, including the following: The training data size is insufficient and does not include sufficient data samples to accurately represent all of the possible input data values.

Now I will modify the hyperparameters in order to get a model that is more reliable and accurate.

```
rfc = RandomForestClassifier()
from sklearn.model_selection import RandomizedSearchCV
# number of trees in random forest
n_estimators = [10,50,100,500,1000]
# number of features at every split
max_features = [‘auto’, ‘sqrt’]
# max depth
max_depth = [int(x) for x in np.linspace(100, 500, num = 11)]
max_depth.append(None)
# create random grid
random_grid = {
 ‘n_estimators’: n_estimators,
 ‘max_features’: max_features,
 ‘max_depth’: max_depth
 }
# Random search of parameters
rfc_random = RandomizedSearchCV(estimator = rfc, param_distributions = 
random_grid,
 n_iter = 100, cv = 3, verbose=2, random_state=42, n_jobs = -1)
# Fit the model
rfc_random.fit(X_train, y_train)
```

Results

Model Evaluation and Validation

During development , a validation set was used to evaluate the model . The final architecture and hyperparameters were chosen because they performed better

Conclusion

- I found this project challenging, mainly due to the structure of the data in thetranscript dataset.

- Majority classes are performing well but the minorities are not.Problem of imbalanceddataset
- Most of the events are wrongly predicted as 'offer received'; offer received is the mostoccuring event or class.
- The main goal I chose was to build something practical the company could use tomake their choices more efficient.
- But the results of the model seem not so good . There is no change in rate of accuracy;it remains constant .

Improvement

- As we’re using an imbalanced dataset we weren't expecting to get a greater accuracy .
- Analysing and building the models was Main challenging part and needed improvement .

Future Work:

- There's a possibility of overfitting, which you can fix by taking into account extra data.
- Build a Machine Learning Pipeline to seamlessly classify new data.
- Make a Machine Learning model to predict transaction amount.
- Deploy Machine Learning model to the web.