Machine Learning Engineer Nanodegree

Capstone Project

Miguel Granica Holgado
October 10st, 2021

I. Definition

Project Overview

- Within the business strategy of the Starbucks company, we focus on the permeability of its offers for a limited group. The datasets provide us with information on both the demographic characteristics of consumers and their receptivity to different types of offers.

- Within the supervised models of Machine learning, these types of problems refer to Classification issues. It consists of a predictive model that infers a target class from a data set

- This model is highly applied in all kinds of disciplines.
    - Email spam detector
    - Conversion prediction
    - Movie review classification
    - MRI images classification
    - Fraud detection
    - e-commerce, customer sentiment analysis…

- We refer to an example that describes a case study of opinion polarity classification:
    - https://www.researchgate.net/publication/328306943_A_Comparison_of_Machine_Learning_Algorithms_in_Opinion_Polarity_Classification_of_Customer_Reviews

- The data is contained in three files:

    - **portfolio.json -** containing offer ids and meta data about each offer (duration, type, etc.)
        - *id* (string) - offer id
        - *offer_type* (string) - type of offer ie BOGO, discount, informational
        - *difficulty* (int) - minimum required spend to complete an offer
        - *reward* (int) - reward given for completing an offer
        - *duration* (int) - time for offer to be open, in days
        - *channels* (list of strings)

- **profile.json** - demographic data for each customer
    - *age* (int) - age of the customer
    - *became_member_on* (int) - date when customer created an app account
    - *gender* (str) - gender of the customer (note some entries contain 'O' for other rather than M or F)
    - *id* (str) - customer id
    - *income* (float) - customer's income

- **transcript.json** - records for transactions, offers received, offers viewed, and offers completed
    - *event* (str) - record description (ie transaction, offer received, offer viewed, etc.)
    - *person* (str) - customer id
    - *time* (int) - time in hours since start of test. The data begins at time t=0
    - *value* - (dict of strings) - either an offer id or transaction amount depending on the record

Problem Statement

- Using the information provided in the datasets, we intend to infer which way a specific client will respond to a certain type of offer.
    - On one hand, We are able to establish a demographic segmentation based on the different categorical fields such as age, gender, income
    - On the other, we can link these demographic segments to the type of offer and their associated receptivity
- The desired objective through this model is to identify the position of the different demographic segments in relation to the offers promoted by the company. And in this way, provide analytical instruments for the future develop a mechanism to optimize the impact of promotional strategies to each of the different clusters.

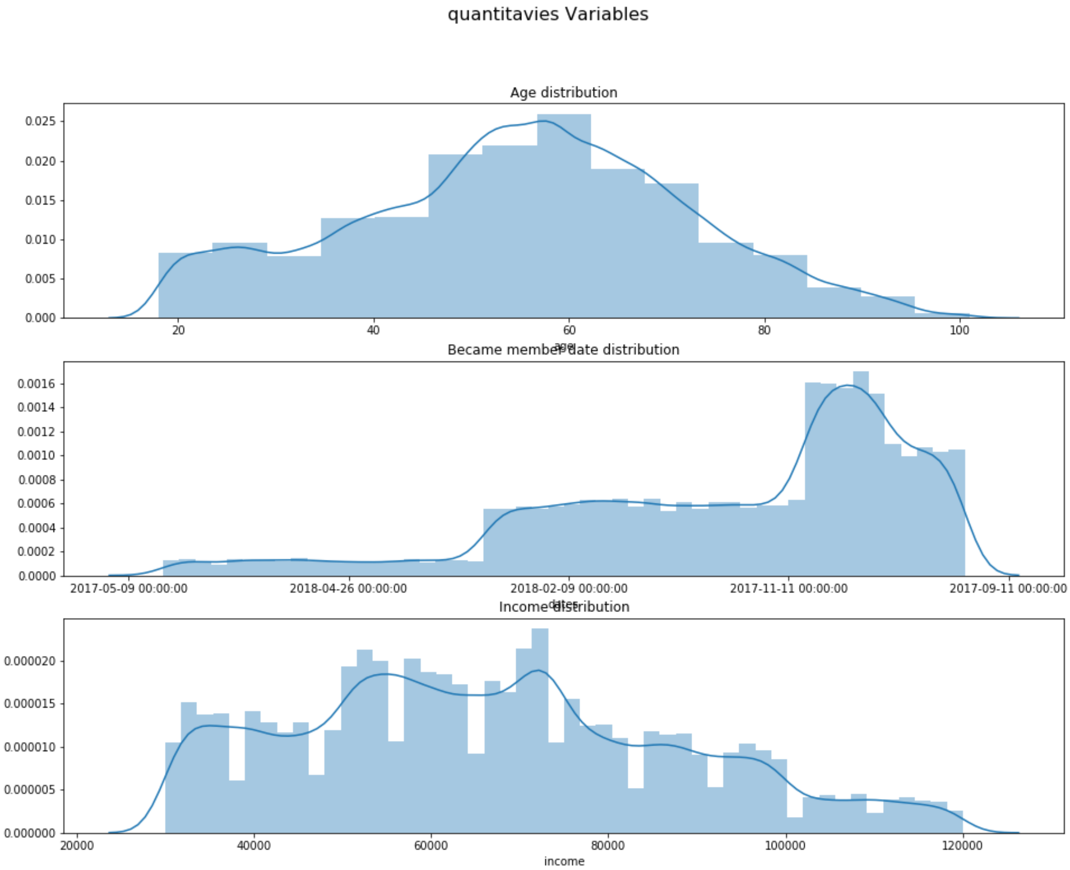
Metrics

- The metrics not only depend on the type of problem we want to solve but also on the distribution of the target class. In this case, the metrics involved are those associated with a Balanced Classification Problem.
    - Accuracy
    - Precision
    - Recall
    - F-Measure

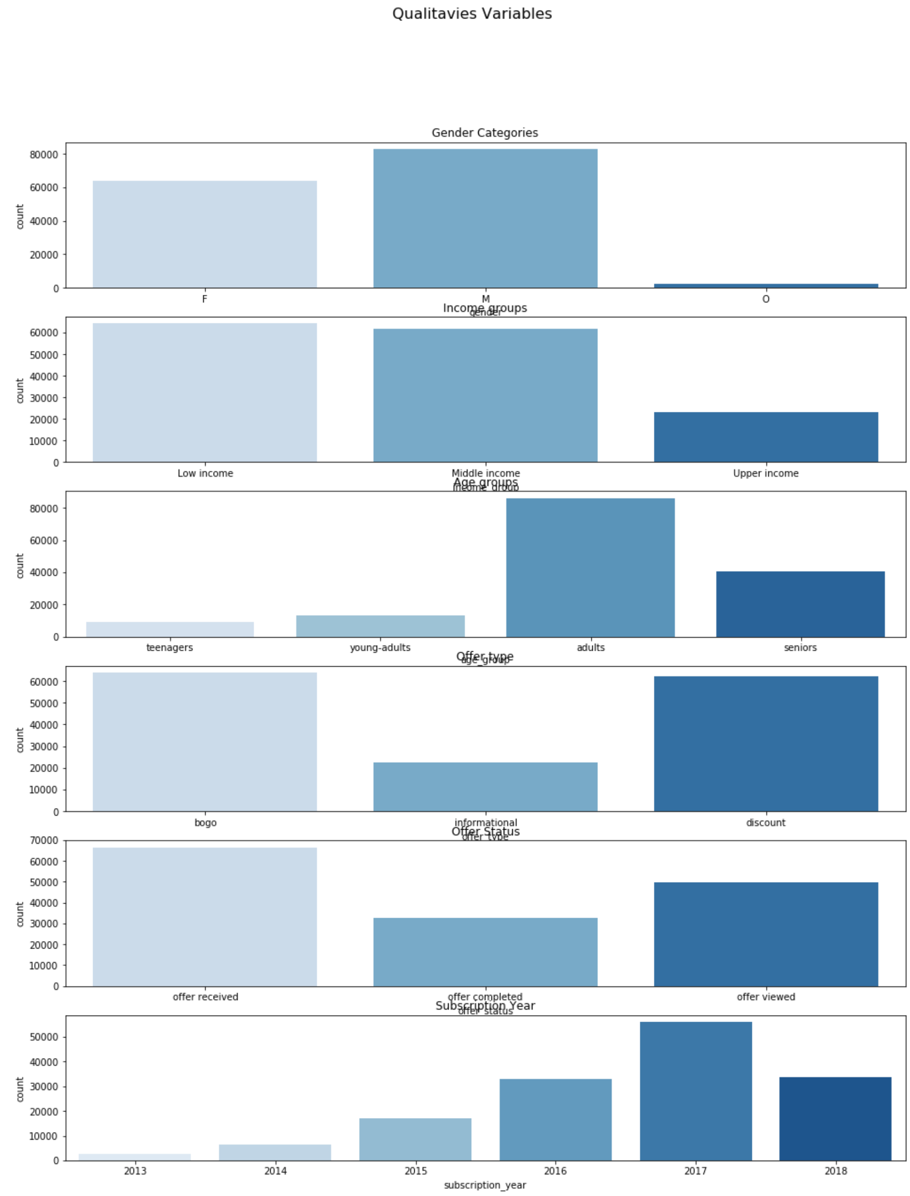
II. Analysis

Data Exploration

- In a first approximation, we focus on obtaining the distributions of the numerical variables:
    - age tends towards a symmetrical distribution with its center around 60 years and with a large sigma
    - the distribution of subscription dates is right skewed, it has its peak at the end of 2018
    - Income is distributed multimodally



- secondly, and to build a general perspective of the features, we analyze the categorical variables:
    - Men with 80,000 interventions are represented 20% more than women. around 2000 cases do not have a defined gender
    - The age group that will consent to the greatest number of cases is that of adults with more than the friendship of the participations. for population segmentation analyzes this bias should be considered
    - with respect to income, upper income has 50% fewer interventions than the other 2
    - the type of offer is divided unevenly: 40 BOGO 40 discount 20 informational
    - the conversion ratio seems less than 50% but we are going to develop it and segment it throughout this notebook
    - There has definitely been a growth in the number of subscribers to strabucks between 2013 and 2018, peaking in 2017



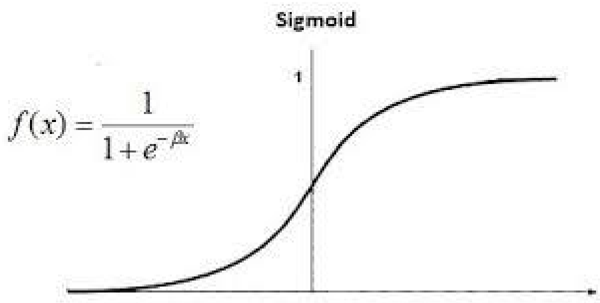
Algorithms and Techniques

- The objective will be to build a machine learning model that allows identifying, based on the different demographic segments, the result of the offers offered to customers.
- To model the predictions about this problem we required supervised Machine learning algorithms. we will use classification algorithms

Logistic regression:

- this model was used in the biological sciences in early twentieth century. It was then adapted by many social science applications.
- is a classification algorithm used to find the probability of event success and event failure.

- It is based on sigmoid function where output is probability and input can be from -infinity to +infinity.



ADVANTAGES	DISVANTAGES
Logistic regression is easier to implement, interpret, and very efficient to train.	If the number of observations is lesser than the number of features, Logistic Regression should not be used, otherwise, it may lead to overfitting.
It makes no assumptions about distributions of classes in feature space.	It constructs linear boundaries.
It can easily extend to multiple classes(multinomial regression) and a natural probabilistic view of class predictions.	The major limitation of Logistic Regression is the assumption of linearity between the dependent variable and the independent variables.
It not only provides a measure of how appropriate a predictor(coefficient size)is, but also its direction of association (positive or negative).	It can only be used to predict discrete functions. Hence, the dependent variable of Logistic Regression is bound to the discrete number set.
It is very fast at classifying unknown records.	Non-linear problems can't be solved with logistic regression because it has a linear decision surface. Linearly separable data is rarely found in real-world scenarios.
Good accuracy for many simple data sets and it performs well when the dataset is linearly separable.	Logistic Regression requires average or no multicollinearity between independent variables.
Logistic regression is less inclined to over-fitting but it can overfit in high dimensional datasets.One may consider Regularization (L1 and L2) techniques to avoid over-fittingin these scenarios.	In Linear Regression independent and dependent variables are related linearly. But Logistic Regression needs that independent variables are linearly related to the log odds (log(p/(1-p)).
It can interpret model coefficients as indicators of feature importance.	It is tough to obtain complex relationships using logistic regression. More powerful and compact algorithms such as Neural Networks can easily outperform this algorithm.

Support Vector Machine

ADVANTAGES	DISVANTAGES
SVM works relatively well when there is a clear margin of separation between classes.	not suitable for large data sets.
more effective in high dimensional spaces.	Does not perform very well when the data set has more noise i.e. target classes are overlapping.
is effective in cases where the number of dimensions is greater than the number of samples.	In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.
is relatively memory efficient	As the support vector classifier works by putting data points, above and below the classifying hyperplane there is no probabilistic explanation for the classification.

K-Nearest Neighbors

- kNN is a lazy learner and non parametric algorithm. It works by assigning a label to an unlabelled point based on the proximity of the unlabelled point to all the other nearest labelled points.
- Lazy Learner implies that it doesn't learn a discriminative function from the training data but rather memorizes the training data instead
- Non-parametric implies that the algorithm makes no assumptions about the distribution of the data.

ADVANTAGES	DISADVANTAGES
easy to understand and implement	As your training data increases, the speed at which calculations are made rapidly decrease
immediately adapt to new training data	Poor performance on imbalanced data
flexibility from the users side to use a distance metric which is best suited for their application (Euclidean, Minkowski, Manhattan distance etc.)	Optimal value of K — If chosen incorrectly, the model will be under or overfitted to the data

Decision Tree

- Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

- Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute as shown in the above figure. This process is then repeated for the subtree rooted at the new node.

ADVANTAGES	DISADVANTAGES
are able to generate understandable rules.	are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.
perform classification without requiring much computation.	are prone to errors in classification problems with many class and relatively small number of training examples.
are able to handle both continuous and categorical variables.	Decision tree can be computationally expensive to train. The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field must be sorted before its best split can be found. In some algorithms, combinations of fields are used and a search must be made for optimal combining weights. Pruning algorithms can also be expensive since many candidate sub-trees must be formed and compared.
provide a clear indication of which fields are most important for prediction or classification.	

random forest

- It builds a forest with an ensemble of decision trees. It is an easy to use machine learning algorithm that produces a great result most of the time even without hyperparameter tuning.

ADVANTAGES	DISVANTAGES
can be used for both classification and regression tasks.	not easily interpretable. They provide feature importance but it does not provide complete visibility into the coefficients as linear regression.
work well with both categorical and numerical data. No scaling or transformation of variables is usually necessary.	can be computationally intensive for large datasets.
implicitly perform feature selection and generate uncorrelated decision trees. It	black box algorithm, you have very little control over what the model does.

does this by choosing a random set of features to build each decision tree. This also makes it a great model when you have to work with a high number of features in the data.	
are not influenced by outliers to a fair degree. It does this by binning the variables.	
can handle linear and non-linear relationships well.	
enerally provide high accuracy and balance the bias-variance trade-off well. Since the model's principle is to average the results across the multiple decision trees it builds, it averages the variance as well.	

Benchmark

- As a baseline solution to compare the results of the final model against. we will use the results of a simplistic model:
    - Logistic Regression model

III. Methodology

Data Preprocessing

- Data Cleaning
    - portfolio dataframe:
        - Rename id column name to offert_id and set as index
        - Hot-encoding the offer_type column
        - Hot-encoding the channels column
    - profile dataframe:
        - Drop age values == 118
        - Drop NaN values for income and gender columns
        - Dateformat became_member_on
        - Binary values for gender column
    - transcript dataframe:
        - unstack amount and offer id columns
        - set time unit in days

- Master table consolidation

	gender	age	income	subscription_year	offer_id	time	amount	reward	difficulty	duration	email	mobile	social	web	offer_type
1	1	55.0	112000.0	2017	1	22	0.0	5.0	5	7	1	1	0	1	1
3	1	75.0	100000.0	2017	1	0	0.0	0.0	5	7	1	1	0	1	1
4	1	75.0	100000.0	2017	1	5	0.0	5.0	5	7	1	1	0	1	1
6	2	68.0	70000.0	2018	1	17	0.0	0.0	5	7	1	1	0	1	1
7	2	68.0	70000.0	2018	1	21	0.0	5.0	5	7	1	1	0	1	1

- Exploratory Data Analysis
    - Univariable exploration
    - Bivariable exploration
    - Segmentation demographics and offer status summary table

- Machine learning preprocessing
    - Based on the different categorical fields, how the different demographic segmentations respond to each one of the types of offers present in the exercise, we can build a Machine learning model that indicates how a certain demographic profile would respond to the different types of offers
    - To validate the impact that Starbucks promotions have, we are going to exclude received offers from the study, in this way we will only take into account the offers seen over the completed ones.
    - Map the categorical fields
        - offer_Status
        - offer_type
        - gender

```
# Build dictionary mappings for Categorical fields
status_map = {'offer completed':1, 'offer viewed':2}
type_map = {'bogo':1, 'informational':2, 'discount':3}
gender_map = {'F':1, 'M':2, 'O':3}
offers_id = df_master['offer_id'].unique().tolist()
```

```
offer_id_map = {value:index for index, value in enumerate(offers_id, start=1)}

# map Offer status with numeric values
df_master['offer_status'] = df_master['offer_status'].map(status_map)
# map Offer type with numeric values
df_master['offer_type'] = df_master['offer_type'].map(type_map)
# map gender with numeric values
df_master['gender'] = df_master['gender'].map(gender_map)
# map offer_id with numeric values
df_master['offer_id'] = df_master['offer_id'].map(offer_id_map)

# Rename 'reward_x' column to 'reward'
df_master.rename(columns ={'reward_x':'reward'}, inplace = True)
```

- Standardize numerical values with MinMaxScaler from scikit learn

	gender	age	income	subscription_year	offer_id	time	amount	reward	difficulty	duration	email	mobile	social	web	offer_type
1	1	0.445783	0.911111	0.8	1	0.758621	0.0	0.5	0.25	0.571429	1	1	0	1	1
3	1	0.686747	0.777778	0.8	1	0.000000	0.0	0.0	0.25	0.571429	1	1	0	1	1
4	1	0.686747	0.777778	0.8	1	0.172414	0.0	0.5	0.25	0.571429	1	1	0	1	1
6	2	0.602410	0.444444	1.0	1	0.586207	0.0	0.0	0.25	0.571429	1	1	0	1	1
7	2	0.602410	0.444444	1.0	1	0.724138	0.0	0.5	0.25	0.571429	1	1	0	1	1

Implementation

- Regarding data cleaning, we have concatenated the unpack of a dictionary contained in a field using a .iteritems within a tuple comprehension to then generate the columns associated with the key-value pair

```
# unpack value column
df_transcript = pd.concat([df_transcript, pd.DataFrame((d for idx, d in df_transcript['value'].iteritems()))], axis=1).fillna(0)
# integrate offer id & offer_id
df_transcript['offer_id'] = np.where(df_transcript['offer id'] == 0, df_transcript['offer_id'], df_transcript['offer id'])
# drop unnecessary columns
df_transcript.drop(columns=['offer id', 'value'], inplace=True)
```

- To carry out the visualization we have built some functions as well

```
def barchar(df, cat_var, order_list=None, rot=None, hue=None):

    plt.figure(figsize=(16,9))
    base_color = sns.color_palette()[0]
    sns.countplot(data= df, x=cat_var, hue=hue, color=base_color, order=order_list)

    #add annotations
```

```
    n_values = df.shape[0]
    cat_count = df[cat_var].value_counts()
    locs, labels = plt.xticks()
    plt.title(f'{cat_var}', fontsize=20)
    plt.xticks(rotation=rot);
```

- In addition, to generate interactive filtering tables we have incorporated the ***ipywidgets*** package that allows you to interactively view different levels of filtering at the same time. We use this tool to build the summary table of the segmented demographics

```
GENDER_ALL = ['F', 'M', 'O']
AGE_GROUP_ALL = ['adults', 'seniors', 'teenagers', 'young-adults']
INCOME_GROUP_ALL = ['Low income', 'Middle income', 'Upper income']
OFFER_TYPE_ALL = ['bogo', 'discount', 'informational']


filtered_output = widgets.Output()

multisel_gender = widgets.SelectMultiple(
                        options=GENDER_ALL,
                        value=GENDER_ALL,
                        #rows=10,
                        description='Gender',
                        disabled=False)

multisel_age_group = widgets.SelectMultiple(
                        options=AGE_GROUP_ALL,
                        value=AGE_GROUP_ALL,
                        description='Age group',
                        disabled=False)

multisel_income_group = widgets.SelectMultiple(
                        options=INCOME_GROUP_ALL,
                        value=INCOME_GROUP_ALL,
                        description='Income',
                        disabled=False)

multisel_offer_type = widgets.SelectMultiple(
                        options=OFFER_TYPE_ALL,
                        value=OFFER_TYPE_ALL,
                        description='Offer type',
                        disabled=False)

def common_filtering(gender, age_group=None, income_group=None,
offer_type=None):
```

```
    global master_filtred
    filtered_output.clear_output()
    if (gender == GENDER_ALL) & (age_group == AGE_GROUP_ALL) &
(income_group == INCOME_GROUP_ALL) & (offer_type ==OFFER_TYPE_ALL):
        master_filtered = master_pivoted

    elif (gender == GENDER_ALL):
        master_filtered =
master_pivoted.loc[(master_pivoted.index.get_level_values('gender').isin
(gender))]

    elif (gender == GENDER_ALL) & (age_group == AGE_GROUP_ALL):
        master_filtered =
master_pivoted.loc[(master_pivoted.index.get_level_values('gender').isin
(gender)) &

(master_pivoted.index.get_level_values('age_group').isin(age_group))]

    elif (gender == GENDER_ALL) & (age_group == AGE_GROUP_ALL) &
(income_group == INCOME_GROUP_ALL):
        master_filtered =
master_pivoted.loc[(master_pivoted.index.get_level_values('gender').isin
(gender)) &

(master_pivoted.index.get_level_values('age_group').isin(age_group)) &

(master_pivoted.index.get_level_values('income_group').isin(income_group
))]

    else:
        master_filtered =
master_pivoted.loc[(master_pivoted.index.get_level_values('gender').isin
(gender)) &

(master_pivoted.index.get_level_values('age_group').isin(age_group)) &

(master_pivoted.index.get_level_values('income_group').isin(income_group
)) &

(master_pivoted.index.get_level_values('offer_type').isin(offer_type))]
    with filtered_output:
        display(master_filtered)

def filter_multisel_gender(change):
    common_filtering(change.new, multisel_age_group.value,
multisel_income_group.value, multisel_offer_type.value)
```

```
def filter_multisel_age_group(change):
    common_filtering(multisel_gender.value, change.new,
multisel_income_group.value, multisel_offer_type.value)

def filter_multisel_income_group(change):
    common_filtering(multisel_gender.value, multisel_age_group.value,
change.new, multisel_offer_type.value)

def filter_multisel_offer_type(change):
    common_filtering(multisel_gender.value, multisel_age_group.value,
multisel_income_group.value, change.new)

multisel_gender.observe(filter_multisel_gender, names='value')
multisel_age_group.observe(filter_multisel_age_group, names='value')
multisel_income_group.observe(filter_multisel_income_group,
names='value')
multisel_offer_type.observe(filter_multisel_offer_type, names='value')


display(multisel_gender)
display(multisel_age_group)
display(multisel_income_group)
display(multisel_offer_type)
```

- Finally we have built 2 functions plus one to train and predict the model and another to evaluate its results

```
def model_train_and_predict(model, X_train=X_train, y_train=y_train):
    # fit the model
    model.fit(X_train, y_train)
    # predict the vañues
    y_pred = model.predict(X_test)

    return y_pred

def model_evaluate(y_true, y_pred):
    #calculate accuracy
    accuracy = accuracy_score(y_true, y_pred)
    print('Accuracy: %.3f' % accuracy)
    # calculate prediction
    precision = precision_score(y_true, y_pred)
    print('Precision: %.3f' % precision)
    # calculate recall
    recall = recall_score(y_true, y_pred)
```

```
print('Recall: %.3f' % recall)
# calculate score
score = f1_score(y_true, y_pred)
print('F-Measure: %.3f' % score)
```

Refinement

- For each one of the studied models, the default parameter settings were used.

IV. Results

Model Evaluation and Validation

- by the evaluation metrics it looks like we have overfitted the model, an useful tool to apply for feature reduction would be the PCA, and with this implementation apply some hyperparameter tuning to sharpen the model

	Decision Tree	**Random Forest**	**Logistic Regression**	**Support Vector Machine**	**Naive Bayes**	**K-Nearest Neighbors**
Accuracy	100	100	100	86.7	100	83.8
Precision	100	100	100	91.6	100	83.1
Recall	100	100	100	72.9	100	73.7
F-Measure	100	100	100	81.2	100	78.1

- The results obtained leave us satisfied in terms of the high accuracy in most of the models executed. In this way, we discard the incorporation of parameter tuning. However, in the future, we consider it interesting to implement a previous PCA to focus the study on the most significant features.

Justification

- Both the benchmark and the other selected models give us very high accuracy results

V. Conclusion

Free-Form Visualization

- Summary table: demographic segmentation of offer status

  We decided to build an interactive dataframe using ipywidgets to have a better understanding of the behaviors that different consumer groups have. For this, we are first going to group the dataframe by the categorical fields that identify demographic groups:

  - gender
  - age
  - income

  and by the fields that describe the characteristics of the established offers:

  - offer type
  - offer status

gender	age_group	income_group	offer_type	offer completed	offer received	offer viewed	offer conversion
F	teenagers	Low income	bogo	191.0	357.0	295.0	53.50
			discount	225.0	341.0	213.0	65.98
			informational	NaN	189.0	125.0	NaN
		Middle income	bogo	104.0	144.0	119.0	72.22
			discount	100.0	149.0	89.0	67.11
			informational	NaN	61.0	37.0	NaN
	young-adults	Low income	bogo	345.0	546.0	446.0	63.19
			discount	334.0	511.0	319.0	65.36
			informational	NaN	293.0	178.0	NaN
		Middle income	bogo	147.0	222.0	178.0	66.22
			discount	137.0	209.0	130.0	65.55
			informational	NaN	103.0	70.0	NaN
	adults	Low income	bogo	1238.0	2027.0	1724.0	61.08
			discount	1373.0	2074.0	1433.0	66.20
			informational	NaN	1018.0	757.0	NaN
		Middle income	bogo	2009.0	2782.0	2366.0	72.21
			discount	2195.0	2879.0	2191.0	76.24
			informational	NaN	1427.0	1069.0	NaN
		Upper income	bogo	1096.0	1412.0	1129.0	77.62
			discount	1094.0	1379.0	966.0	79.33
			informational	NaN	729.0	488.0	NaN

Reflection

- In the first instance we concentrate on structuring the data sets, understanding the information that they do not provide and doing the pertinent data wrangling necessary to continue with the EDA. These intermediate transformations added to the analysis and their respective visualizations gave us a more acute perspective of the characteristics of the datasets and their possibilities. In summary, we decided to implement an interactive table that would describe the convertibility ratio between the different types of offer for each of the demographic segments. This descriptive analysis gave us a very complete picture regarding the diversity of results found. Finally, we convert the datasets into a master table which we use to build the machine learning models.
- The fact of being able to be working and analyzing business data in a study environment is highly valuable and makes the final project very attractive. Here the skills developed throughout the course should be applied as a means to provide a solution to a specific client. It was a very complete training for me and I had a lot of fun reflecting on the insights that could be extracted from this data

Improvement

- Training a multiclass classification model is a potential implementation that can give us a more robust perspective on the effectiveness of these offers.
- On the other hand, using the Xgboost model may be a good possibility also due to its robust predictions and its easy implementation aswell.
- Test other ML algorithms like neural networks to make classifications.