1. Introduction

A retail company "ABC Private Limited" wants to understand the customer purchase behavior (specifically, purchase amount) against various products of different categories. They have shared purchase summary of various customers for a selected high volume products from last month. They want to build a model to predict the purchase amount of customer against various products which will help them to create personalized offer for customers against different products.

1.1 Purpose

1.1.1 Frame the problem

Before looking at the data it is important to understand how does the company expect to use and benefit from this model? This first brainstorming helps to determine how to frame the problem, what algorithms to select and measure the performance of each one.
We can categorize our Machine Learning (ML) system as:

- Supervised Learning task: we are given *labeled* training data (e.g. we already know how much a customer spent on a specific product)
- Regression task: our algorithm is expected to predict the purchase amount a client is expected to spend on this day.
- Plain batch learning: since there is no continuous flow of data coming into our system, there is no particular need to adjust to changing data rapidly, and the data is small enough to fit in memory, so plain batch learning should work.

If you want to learn more on how to categories a ML system you can read

1.1.2 Select a Performance Measure

Usually for regression problems the typical performance measure is the Root Mean Square Error (RMSE). This function gives an idea of how much error the system makes in its predictions with higher weight for large errors.
Make Assumptions
Before even looking at the available data it is good to make some assumption on the expected results. Therefore, let's start to think about possible parameters that might influence the amount a client spends on Black Friday.

1.2 Scope of the project

1.2.1 Initial functional requirement will be:-

- Selecting the algorithm meeting requirement.
- Choosing the optimum algorithm form set of algorithm.
- Testing it on the datasets.
- After getting the result if the result is low change the hyper parameters.
- Out of all result get best of all.

1.2.2 Initial non functional requirement will be:-

- Getting the large datasets which can provide developer enough image to train the model.
- Maintain the minimum variance& bias so the model is successfully work.
- Avoid the underfitting and overfitting.

Terms	Definition
Dataset	Data for training and testing for the model
Variance	Difference between the training and testing accuracy
Bias	Both learning and training accuracy is low
Overfitting	Model is very complex
Underfitting	Model is bias
Developer	Who is developing the model
Review	A written recommendation about the appropriateness of an Product for selling and buying may include suggestions for improvement.
Reviewer	A person that examines an Product and has the ability to recommend approval Product for buying or to request that changes be made in the Product.
Software Requirement Specification	A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document
User	Reviewer

1.3 References

- https://datahack.analyticsvidhya.com/contest/black-friday/
- Black Friday (n.d.). About.com: retail industry located at http://retailindustry.about.com/od/abouttheretailindustry/g/black_friday.htm
- Barbaro, M. (2006, November 25). Attention, holiday shoppers: We have fisticuffs in aisle 2 [Electronic version]. The New York Times Late Edition. Retrieved June 11, 2007 from LexisNexis Academic database.

1.4 Software Life Cycle Model:

In order to make this Project we are going to use Classic LIFE CYCLE MODEL .Classic life cycle model is also known as WATER FALL MODEL. The life cycle model demands a Systematic sequential approach to software development that begins at the system level and progress through analysis design coding, testing and maintenance

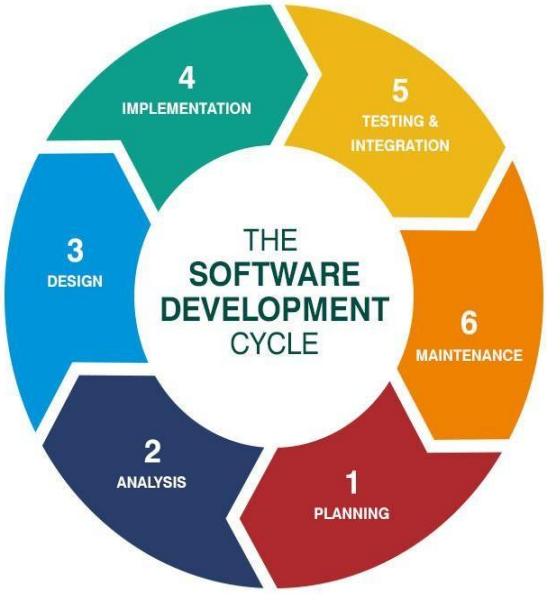

Fig 1.4.1 Software life cycle model

The Classic Life Cycle Model

The waterfall model is sequential software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of conception initiation, Analysis, Design (validation), construction. Testing and maintained.

1) System Engineering and Analysis:

Because software is always a part of larger system work. Begins by establishing requirement for all system elements and Then allocating some subset of these requirement to the software system Engineering and analysis encompasses the requirement gathering at the system level with a small amount of top level design and analysis.

2) Software requirement Analysis:

The requirement gathering process is intensified and focused specifically on the software Requirement for the both system and software are discussed and reviewed with the customer. The customer specifies the entire requirement or the software and the function to be performed by the software.

3) Design:

Software design is actually a multi-step process that focuses on distinct attributes of the program data structure, software architecture, procedural detail and interface of the software that can be assessed or quality before coding begins .Like requirement the design is documented and becomes part of the software.

4) Coding:

The design must be translated into a machine readable form. The coding step performs this task. If design is programmed in a detailed manner, coding can be accomplished mechanically.

5) Testing:

Once code has been generated programmed testing begins. The testing process focuses on the internals of the software ensuring that all statement have been tested and on the functional externals hat is conducting tests to uncover the errors and ensure that defined input will produce the results that agree with the required results.

Unit testing:

In computer programming, Unit testing is software Verification and validation method where the programmer gains confidence that individual units of source code are fit to use A unit is the smallest testable part of an application. In procedural programming a unit may be an individual programmed, function, procedure, etc. while in object-oriented programming, the smallest Unit is a class, which may belong to a base/super class abstract class or derived/child class.

Benefits:

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict written contract that the piece of code must satisfy.

Documentation:

Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit and how to use it can look at the tests to gain a basic understanding of the unit API.

Limitation of unit testing:

Testing cannot be expected to catch error in the program –It is impossible to evaluate all execution paths for all but the most trivial programs. The same is true for unit testing. Additionally, by unit testing only types the functionality if the units themselves.

6) Maintenance:

Software will undoubtedly undergo change after it is Delivered to the customer .Change will occur because errors have been encountered because the software must be able adopted to accommodate changes in its external environment because the customer requires functional or performance enhancement enhancements. The classic life cycle is the oldest and most widely used paradigm or software engineering

2. Overall Decision

2.1 Factors affecting purchase Method Life Cycle

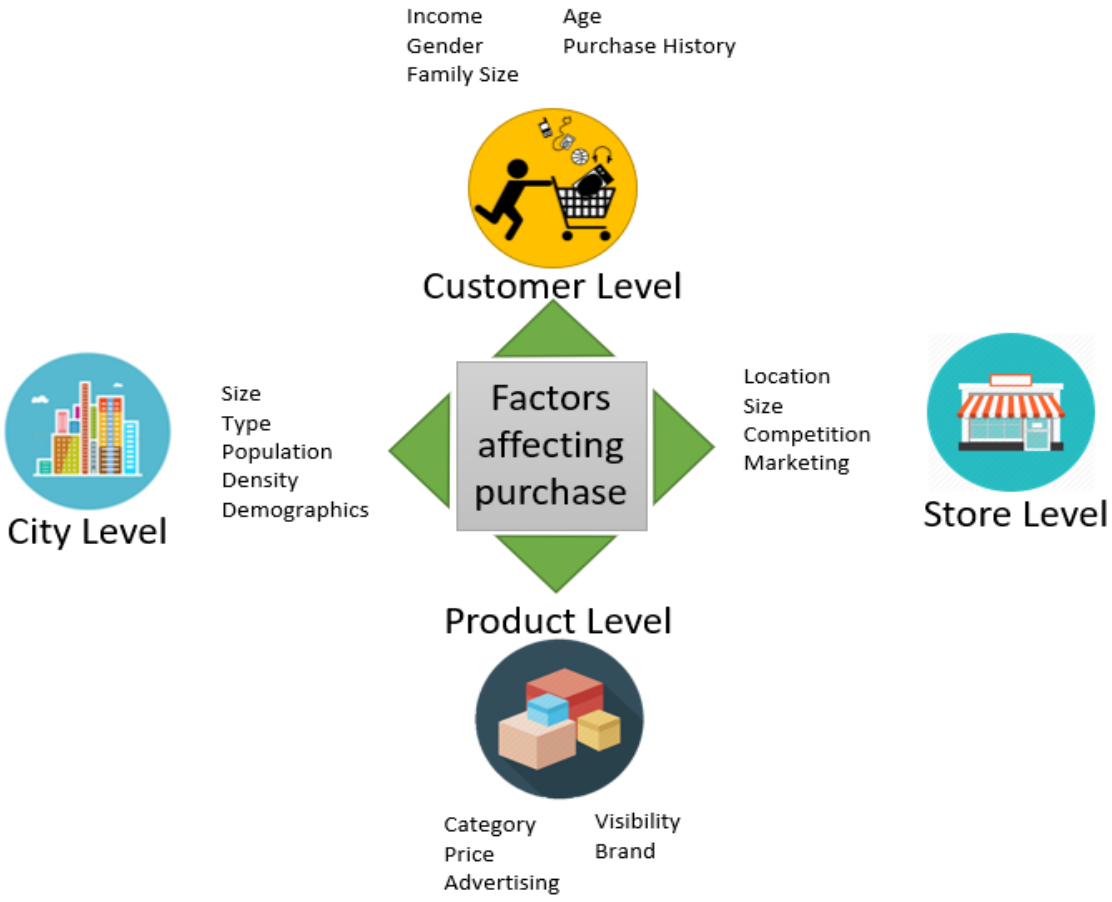

Fig 2.1.1 Factors affecting Purchase

City Level Hypotheses:

1. City Type and Size: Urban or Tier 1 cities should have higher sales because of the higher income levels of people there.
2. Population Density: Cities with densely populated areas should have higher sales because of more demand.
3. Younger Population : Cities with younger populations might have higher tendency to spend more on Black Friday

Customer Level Hypotheses:

1. Income: People with higher income should spend more on products.
2. Age and Gender: Men with ages ranging from 25 to 40 should spend more on technological products.
3. Family Size: Families should be more contained on spending; just buying the best offers and only needed products.

4. Purchase History: Customer with a purchase history should be more willing to purchase more products on this day.

Store Level Hypotheses:

1. Location: Stores with a location in well moved streets should have better sales.
2. Size: Bigger stores with higher stores and variety of products should have better sales.
3. Competition: Stores with no competitors nearby must have the highest sales.
4. Marketing: Do stores which spend more on marketing should have the best sales results

Product Level Hypotheses:

1. Category: Most clients should be looking to buy technological products;
2. Price: Customer will spend more on products with higher discounts
3. Advertising: More advertised products should sell more
4. Visibility: More visible products should sell more
5. Brand: Clients will invest more on already known brands

2.2 Available data

This is the current data they have available:

Data

Variable	Definition
User_ID	User ID
Product_ID	Product ID
Gender	Sex of User
Age	Age in bins
Occupation	Occupation (Masked)
City_Category	Category of the City (A,B,C)
Stay_In_Current_City_Years	Number of years stay in current city
Marital_Status	Marital Status
Product_Category_1	Product Category (Masked)
Product_Category_2	Product may belongs to other category also (Masked)
Product_Category_3	Product may belongs to other category also (Masked)
Purchase	Purchase Amount (Target Variable)

Fig 2.2.1 Available data

Data made available “ABC Private Limited”

If we analyze it individually we see that we do not have any information regarding the stores. Moreover, there is some information related to the customer such as age group, sex, occupation and marital status. On the other hand, we have data on the city's size and how many years the customer has lived in it whereas on the product's side there is only information regarding the categories and the amount spent. It is my belief that Gender, Age, City_Category, Product_Category_1 are the predictors that will influence more the amount spent by a customer on this day.
The target variable is Purchase.

Name	Type	Subtype	Description	Segment	Expectation
User_ID	Numeric	Discrete	User ID	Customer	Low_Impact
Product_ID	Numeric	Discrete	Product ID	Product	Low_Impact
Gender	Categorical	Nominal	Sex of User	Customer	High_Impact
Age	Categorical	Ordinal	Age in bins	Customer	High_Impact
Occupation	Categorical	Nominal	Occupation (Masked)	Customer	Medium_Impact
City_Category	Categorical	Ordinal	Category of the city (A,B,C)	City	High_Impact
Stay_In_Current_City_Years	Categorical	Ordinal	Number of years stay in current city	City	Low_Impact
Marital_Status	Categorical	Ordinal	Marital Status	Customer	Low_Impact
Product_Category_1	Categorical	Nominal	Product Category (Masked)	Product	High_Impact
Product_Category_2	Categorical	Nominal	Product may belongs to other category also (Masked)	Product	Low_Impact
Product_Category_3	Categorical	Nominal	Product may belongs to other category also (Masked)	Product	Low_Impact
Purchase	Numeric	Continuous	Purchase Amount (Target Variable)	Product	NAN

Fig 2.2.2 Description of Data

Take a quick look at the Data Structure

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	3	NaN	NaN	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	14.0	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	NaN	NaN	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	NaN	1057
4	1000002	P00285442	M	55+	16	C	4+	0	8	NaN	NaN	7969

Fig 2.2.2 First five lines of dataset

The above line gives us part of a dataset consisting of 5 rows and all columns.

Age: should be treated as numerical. It presents age groups.
City Category: We can convert this to numerical as well, with dummy variables. Should take a look at the frequency of the values.
Occupation: It seems like it has at least 16 different values, should see frequency and try to decrease this value.
Gender: There are possibly two gender, we can make this binary.
Product_ID: Should see if the string "P" means something and if there are other values.

Stay_In_Current_City_Years: We should deal with the '+' symbol.
Product_Category_2 and Product_Category_3: Have NaN values.
New variables to have in consideration:
User_Count: There are duplicate User_ID, so it would be a good idea to create a feature with number of observations of the user
Product_Count: Number of observations of the product

3. Requirement Specification

3.1 External requirement specification:

The only link to an external system is the link to the image with customer needed to identity. The Administrator believes that a site member is much more likely to be an effective reviewer and has imposed a membership requirement for a Reviewer.

3.2 Detailed Non-Functional Requirements:

3.2.1 Functional Requirement:

First of all model should be successfully trained by developer. Installing Anaconda on Windows

1. Download and install Anaconda (windows version).
2. Select the default options when prompted during the installation of Anaconda.

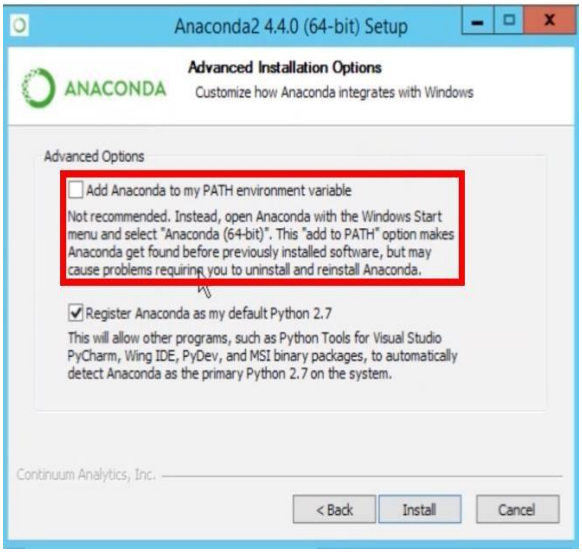

Fig 3.2.1 Installation of Anaconda

3. After you finished installing, open **Anaconda Prompt**. Type the command below to see that you can use a Jupyter (IPython) Notebook.

4. If you didn't check the add Anaconda to path argument during the installation process, you will have to add python and conda to your environment variables. You know you

need to do so if you open a **command prompt** (not anaconda prompt) and get the following messages.

5. This step gives two options for adding python and conda to your path .If you don't know where your conda and/or python is, you type the following commands into your **anaconda prompt.**

3.2.2 Hardware Requirement:

➢ PROCESSOR : I3

➢ RAM : 4GB

➢ OS : WIN7

➢ MONITOR : color

3.2.3 Software Requirement: Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, macOS and Linux.

To get Navigator, get the Navigator cheat sheet and install Anaconda.

The Navigator Getting started with Navigator section shows how to start Navigator from the shortcuts or from a terminal window.

Following libraries of python should be install:

- Numpy library
- Pandas Library
- Matplotlib library

Numpy Library

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy.

Pandas Library

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including

finance, economics, Statistics, analytics, etc. In this tutorial, we will learn the various features of Python Pandas and how to use them in practice.

Matplotlib library

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

4. Training Process of Classification Model

Organization of our analysis

Our goal as a Data Scientist is to identify the most important variables and to define the best regression model for predicting out target variable. Hence, this analysis will be divided into five stages:

1. Exploratory data analysis (EDA);
2. Data Pre-processing;
3. Feature engineering;
4. Feature Transformation;
5. Modeling;
6. Hyper parameter tuning
7. Assembling

4.1. Exploratory Data Analysis (EDA)

We've made our first assumptions on the data and now we are ready to perform some basic data exploration and come up with some inference. Hence, the goal for this section is to take a glimpse on the data as well as any irregularities so that we can correct on the next section, Data Pre-Processing.

4.1.1. Univariate Analysis

To get an idea of the distribution of numerical variables, histograms are an excellent starting point. Let's begin by generating one for Purchase, our target variable.

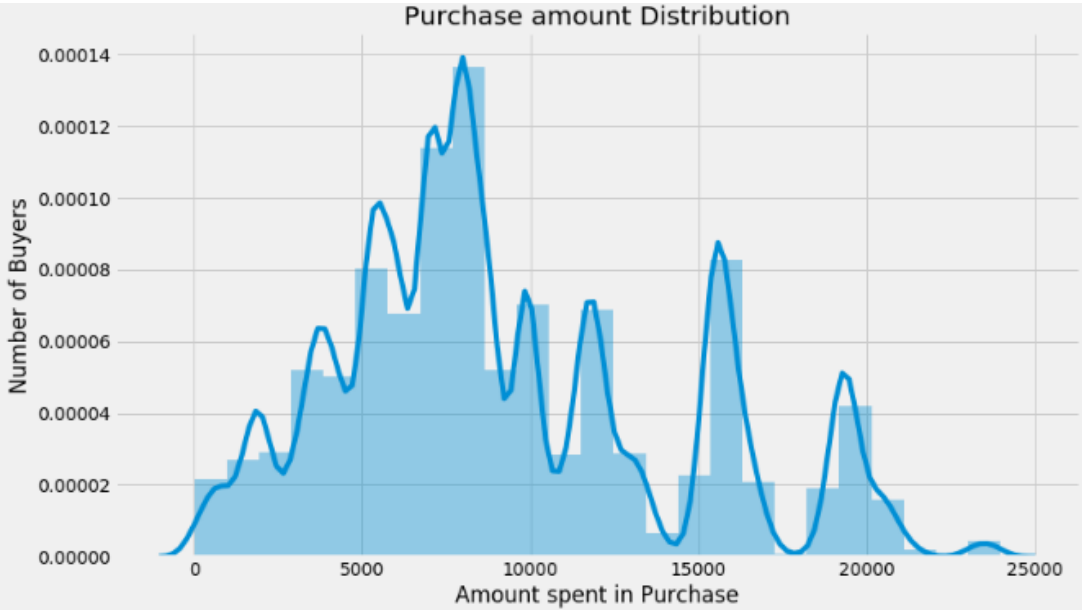

Fig 4.1.1.1 Distribution of the target variable: Purchase

It seems like our target variable has an almost Gaussian distribution.

```
Skew is: 0.6001400037087128
Kurtosis: -0.338378
```

4.1.2. Numerical Predictors

Now that we've analysed our target variable, let's consider our predictors. Let's start by seeing which of our features numeric are.

```
User_ID                    int64
Occupation                 int64
Marital_Status             int64
Product_Category_1         int64
Product_Category_2       float64
Product_Category_3       float64
Purchase                   int64
dtype: object
```

Fig 4.1.2 Numerical Predictors

4.1.2.1. Distribution of the variable Occupation

As seen in the beginning, Occupation has at least 20 different values. Since we do not known to each occupation each number corresponds, is difficult to make any analysis. Furthermore, it seems we have no alternative but to use since there is no way to reduce this number as we did on Project Bigmart with Item_Type.

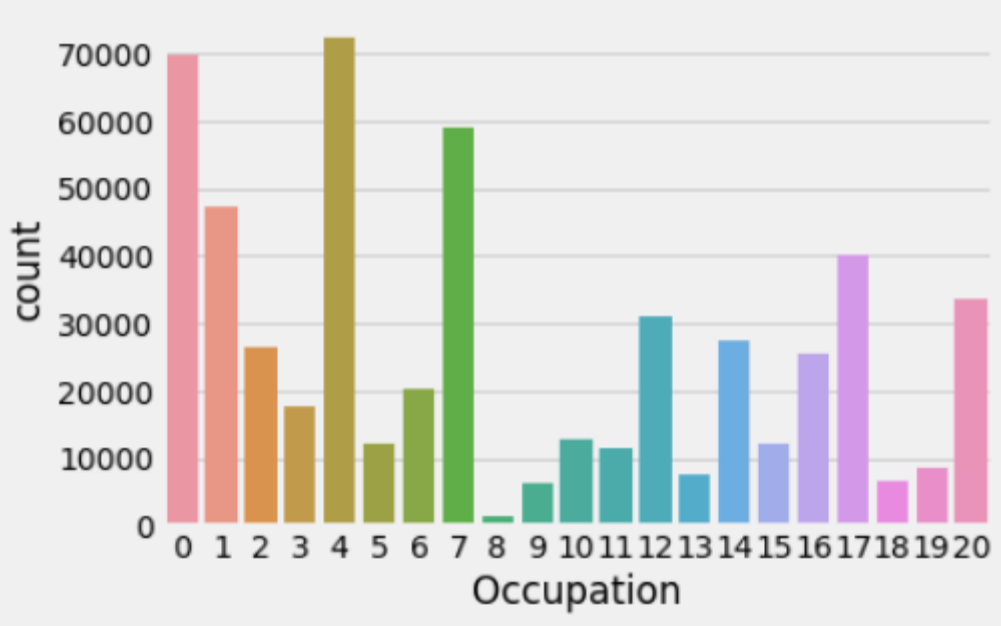

Fig 4.1.2.1 Distribution of the variable Occupation

4.1.2.2. Distribution of the variable Marital_Status

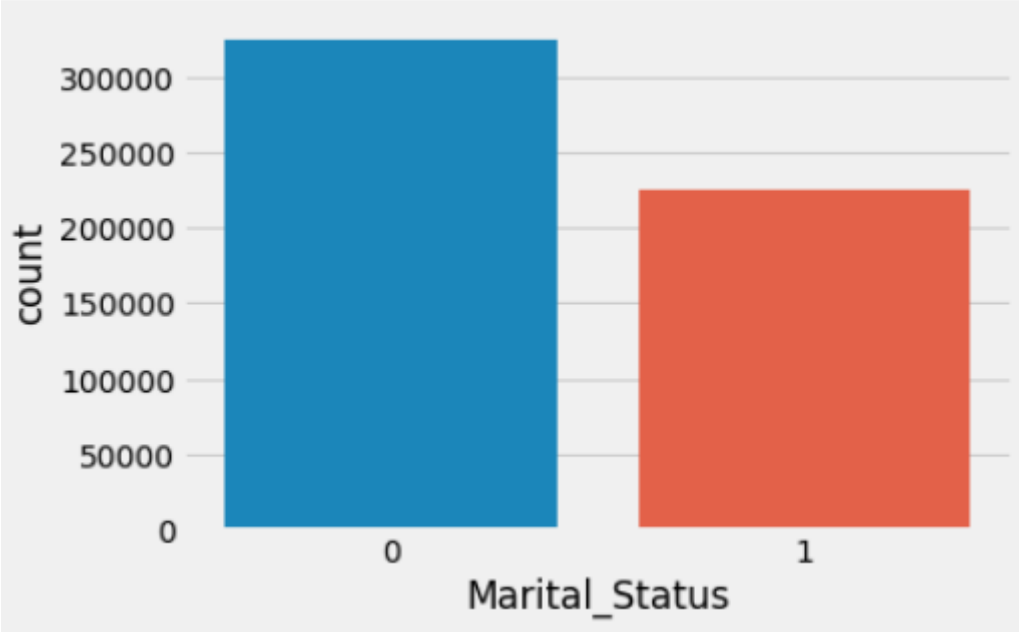

Fig 4.1.2.2 Distribution of the variable Marital_Status

As expected there are more single people buying products on Black Friday than married people, but do they spend more?

4.1.2.3. Distribution of the variable Product_Category_1

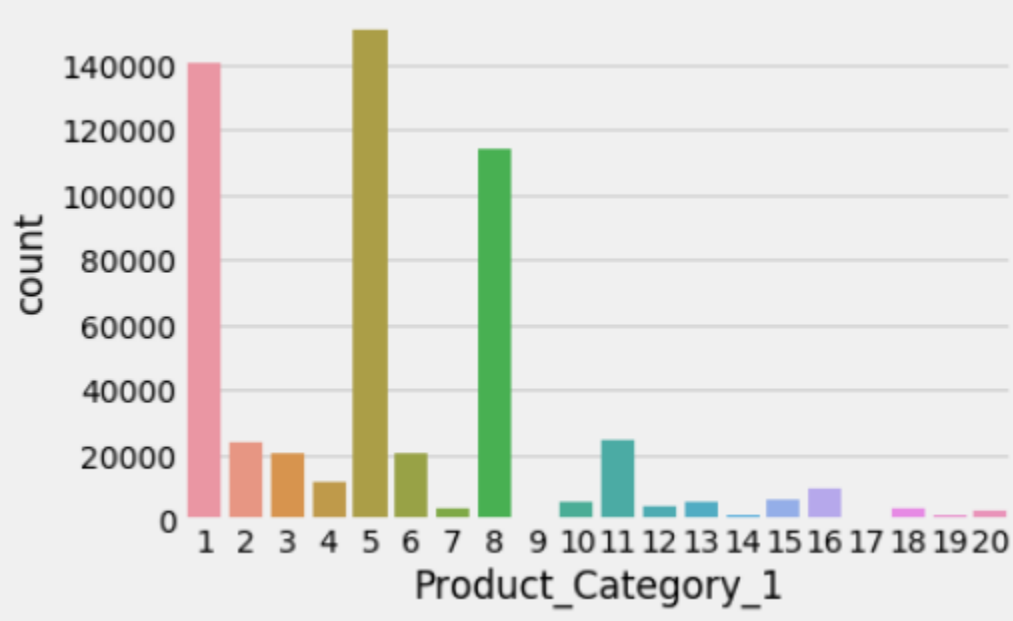

Fig 4.1.2.3 Distribution of the variable Product_Category_1

From the distribution for products from category one, it is clear that three products stand out, number 1, 5 and 8. Unfortunately, we do not know which product each number represents.

4.1.2.4. Correlation between Numerical Predictors and Target variable

```
Purchase                 1.000000
Occupation               0.020833
User_ID                  0.004716
Marital_Status          -0.000463
Product_Category_3      -0.022006
Product_Category_2      -0.209918
Product_Category_1      -0.343703
Name: Purchase, dtype: float64

Purchase                 1.000000
Occupation               0.020833
User_ID                  0.004716
Marital_Status          -0.000463
Product_Category_3      -0.022006
Product_Category_2      -0.209918
Product_Category_1      -0.343703
Name: Purchase, dtype: float64
```

Fig 4.1.2.4 Correlation between Numerical Predictors and Target variable

There does not seem to be any predictor that would have a high impact on Purchase , since the highest correlation is give by Occupation with 0.0208. On the other hand, Product_Category_1 has a negative correlation with our target with the value -0.3437 which is somehow odd.

4.1.2.5 Distribution of the variable Gender

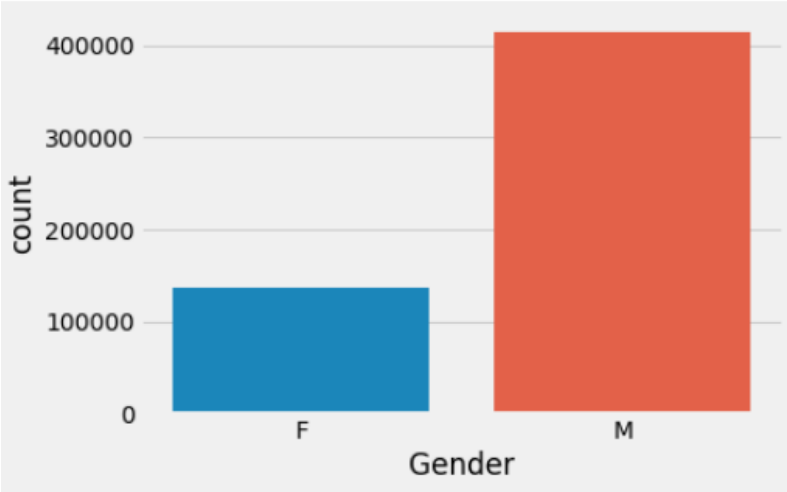

Fig 4.1.2.5 Distribution of the variable Gender

Most of the buyers are males, but who spends more on each purchase: man or woman?

4.1.2.6 Distribution of the variable Age

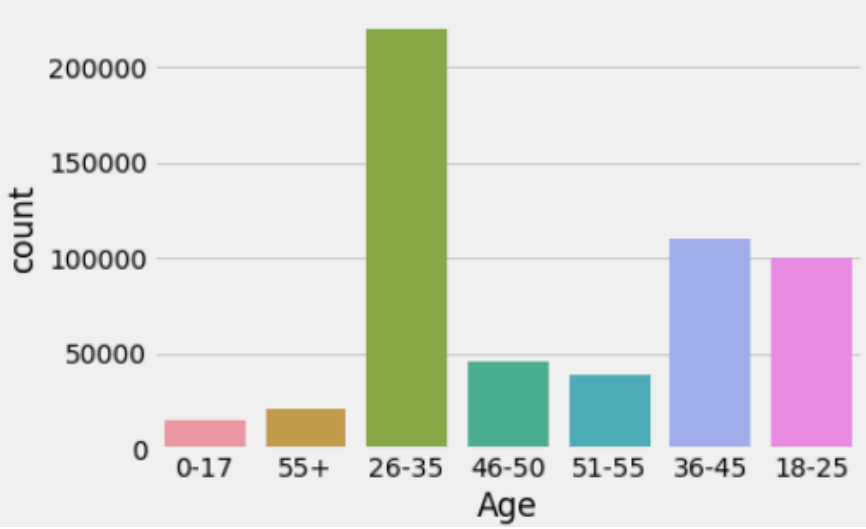

Fig 4.1.2.6 Distribution of the variable Age

As expected, most purchases are made by people between 18 to 45 years old.

4.1.2.7 Distribution of the variable City_Category

Supposing 'A' represents the biggest city whereas 'C' the smallest, it curious to see that the medium size cities 'B' had higher sales than the others. But do they also spent more?

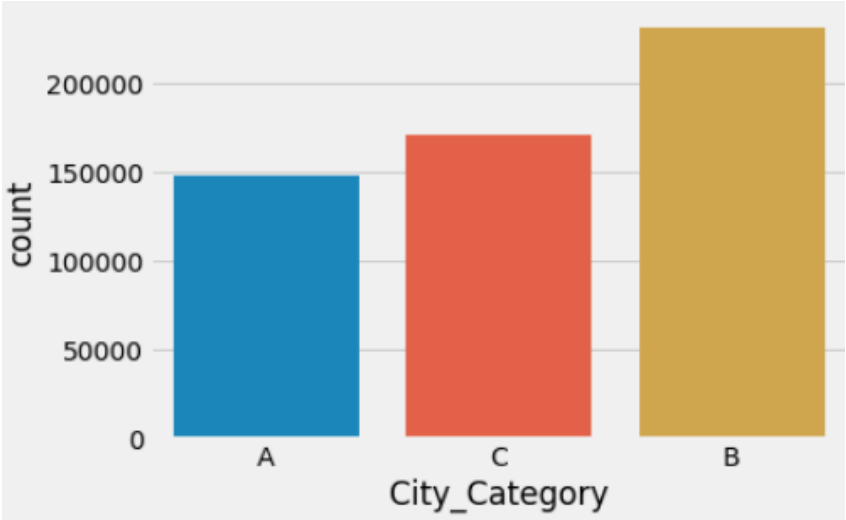

Fig 4.1.2.7 Distribution of the variable City_Category

4.1.2.8 Distribution of the variable Stay_In_Current_City_Years

The tendency looks like the longest someone is living in that city the less prone they are to buy new things. Hence, if someone is new in town and needs a great number of new things for their house that they'll take advantage of the low prices in Black Friday to purchase all the things needed.

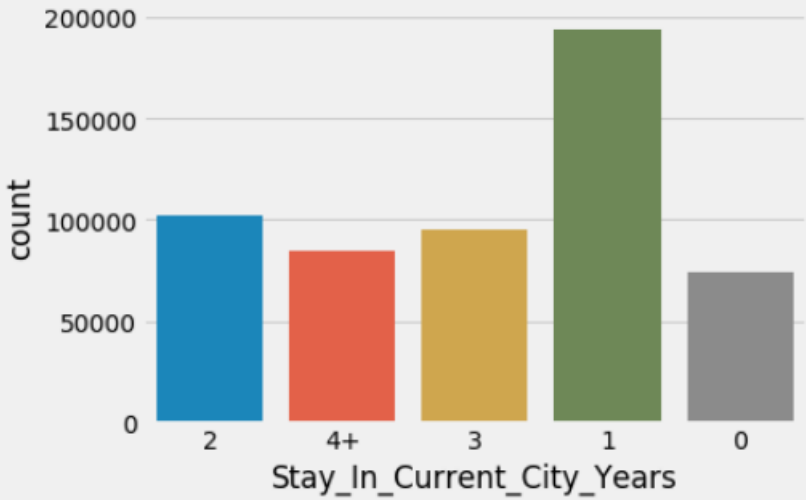

Fig 4.1.2.8 Distribution of the variable Stay_In_Current_City_Years

4.2 Data Pre-Processing

During our EDA we were able to take some conclusions regarding our first assumptions and the available data:

- Age: should be treated as numerical. It presents age groups.
- City_Category: We can convert this to numerical as well, with dummy variables. Should take a look at the frequency of the values.
- Occupation: It seems like it has at least 16 different values, should see frequency and try to decrease this value.
- Gender: There are possibly two gender, we can make this binary.
- Product_ID: Should see if the string "P" means something and if there are other values.
- Stay_In_Current_City_Years: We should deal with the '+' symbol.
- Product_Category_2 and Product_Category_3: Have NaN values.

Usually, datasets for every challenge such as those presented in Analytics Vidhya or Kaggle come separated as a train.csv and a test.csv. It is generally a good idea to combine both sets into one, in order to perform data cleaning and feature engineering and later divide them again. With this step we do not have to go through the trouble of repeting twice the same code, for both datasets. Let's combine them into a dataframe datawith a source column specifying where each observation belongs.

4.2.1 Looking for missing values

```
User_ID                       0.000000
Product_ID                    0.000000
Gender                        0.000000
Age                           0.000000
Occupation                    0.000000
City_Category                 0.000000
Stay_In_Current_City_Years    0.000000
Marital_Status                0.000000
Product_Category_1            0.000000
Product_Category_2           31.388587
Product_Category_3           69.648078
Purchase                     29.808452
source                        0.000000
dtype: float64
```

Fig 4.2.1 Looking for missing values

The only predictors having missing value are Product_Category_1 and Product_Category_2. We can either try to impute the missing values or drop these predictors. We can text both approaches to see which returns the best results.

4.2.1.1. Numerical Values

```
-2.0     245982
 2.0      70498
 3.0       4123
 4.0      36705
 5.0      37165
 6.0      23575
 7.0        854
 8.0      91317
 9.0       8177
 10.0      4420
 11.0     20230
 12.0      7801
 13.0     15054
 14.0     78834
 15.0     54114
 16.0     61687
 17.0     19104
 18.0      4027
Name: Product_Category_2, dtype: int64
```

Fig 4.2.1.1.1 Imputing the value Zero

```
data[“Product_Category_3”]=\
data[“Product_Category_3”].fillna(-2.0).astype(“float”)
data.Product_Category_3.value_counts().sort_index()
```

4.2.1.2. Categorical Values

Apply function len(unique()) to every data variable

```
User_ID                      5891
Product_ID                   3672
Gender                          2
Age                             7
Occupation                     21
City_Category                   3
Stay_In_Current_City_Years      5
Marital_Status                  2
Product_Category_1             18
Product_Category_2             18
Product_Category_3             16
Purchase                    17996
source                          2
dtype: int64
```

Fig 4.2.1.2 Categorical Values

4.2.2 Frequency Analysis

Filter categorical variables and get dataframe will all strings columns names except Item_identfier and outlet_identifier frequency of categories Number of times each value appears in the column

```
This is the frequency distribution for Product_ID:
P00265242    2709
P00025442    2310
P00110742    2292
P00112142    2279
P00046742    2084
P00057642    2079
P00184942    2066
P00058042    2014
P00237542    1999
P00145042    1991
P00117942    1985
P00110942    1976
P00059442    1949
P00010742    1922
P00255842    1914
P00110842    1834
P00220442    1833
P00051442    1794
P00117442    1785
P00102642    1782
P00242742    1774
P00148642    1758
P00031042    1737
P00278642    1735
P00080342    1719
P00028842    1713
P00034742    1690
P00251242    1682
P00114942    1673
P00000142    1636
             ...
P00059342       1
P00081642       1
P00057842       1
P00236842       1
P00315342       1
P00167342       1
P00301442       1
P00322642       1
P00143242       1
P00091542       1
P00069742       1
P00082142       1
P00068342       1
P00065542       1
P00062442       1
P00038642       1
P00083542       1
P00065142       1
P00348142       1
P00260742       1
P00012342       1
P00329842       1
P00224642       1
P00062242       1
P00030342       1
P00185942       1
P00013542       1
P00056342       1
P00203942       1
P00239542       1
Name: Product_ID, Length: 3672, dtype: int64
```

Fig 4.2.2.1 Frequency Analysis

```
This is the frequency distribution for Gender:
M    587052
F    192462
Name: Gender, dtype: int64

This is the frequency distribution for Age:
26-35    311554
36-45    155898
18-25    141209
46-50     64902
51-55     54450
55+       30316
0-17      21185
Name: Age, dtype: int64

This is the frequency distribution for City_Categc
B    328524
C    241487
A    209503
Name: City_Category, dtype: int64

This is the frequency distribution for Stay_In_Cur
1     274937
2     144599
3     134750
4+    120054
0     105174
Name: Stay_In_Current_City_Years, dtype: int64
```

Fig 4.2.2.2 Frequency Description

4.3Feature Engineering

4.3.1. Converting Gender to binary

```
1    587052
0    192462
Name: Gender, dtype: int64
```

Fig 4.3.1 Converting Gender to binary

4.3.2. Converting Age to numeric values

```
2    311554
3    155898
1    141209
4     64902
5     54450
6     30316
0     21185
Name: Age, dtype: int64
```

Fig 4.3.2. Converting Age to numeric values

4.3.3. Converting City_Category to binary

```
1    328524
2    241487
0    209503
Name: City_Category, dtype: int64
```

Fig 4.3.3 Converting City_Category to binary

4.3.4. Converting Stay_In_Current_City_Years to binary

```
User_ID                          int64
Product_ID                      object
Gender                           int64
Age                              int64
Occupation                       int64
City_Category                    int64
Marital_Status                   int64
Product_Category_1               int64
Product_Category_2             float64
Product_Category_3             float64
Purchase                       float64
source                          object
Stay_In_Current_City_Years_0     uint8
Stay_In_Current_City_Years_1     uint8
Stay_In_Current_City_Years_2     uint8
Stay_In_Current_City_Years_3     uint8
Stay_In_Current_City_Years_4     uint8
dtype: object
```

Fig 4.3.4 Converting Stay_In_Current_City_Years to binary

4.3.5. Function to create count features

feature representing the count of each user

Creating all the features:

4.4. Model Building

train_df = pd.read_csv('data/train_modified.csv')

Since I'll be making many models, instead of repeating the codes again and again, I would like to define a generic function which takes the algorithm and data as input and makes the model, performs cross-validation and generates submission.

4.4.1. Linear Regression Model

Regression is to examine two things: (1) does a set of predictor variables do a good job of predicting an outcome (dependent) variable? (2) Which variables, in particular, are significant predictors of the outcome variable, and in what way do they–indicated by the magnitude and sign of the beta estimates–impact the outcome variable? These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables. The simplest form of the regression equation with one dependent and one independent variable is defined by the formula y = c + b*x, where y = estimated dependent variable score, c = constant, b = regression coefficient, and x = score on the independent variable.

4.4.2. Decision Tree Model

A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values. Decision tree builds models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested. Leaf node (e.g., Hours Played) represents a decision on the numerical target. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. If the maximum depth of the tree is set too high, the decision trees learn too fine details of the training data and learn from the noise, i.e. they overfit.

4.4.3. Random Forrest Model

A random forest is an estimator that fits a number of classifying decision trees on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True. The idea behind this technique is to decorrelate the several trees. It generates on the different bootstrapped samples(i.e. self-generated samples) from training Data. And then we reduce the Variance in the Trees by averaging them. Hence, in this approach, it creates a large number of decision trees in python or R. The random forest model is very good at handling tabular data with numerical features, or categorical features with fewer than hundreds of categories. Random forests have the ability to capture the non-linear interaction between the features and the target.

4.4.4. XGBoost Model

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods

do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. Boosting can be interpreted as an optimization algorithm on a suitable cost function. The latter two papers introduced the view of boosting algorithms as iterative functional gradient descent algorithms. That is, algorithms that optimizes a cost function over function space by iteratively choosing a function (weak hypothesis) that points in the negative gradient direction. This functional gradient view of boosting has led to the development of boosting algorithms in many areas of machine learning and statistics beyond regression and classification. Gradient boosting combines weak "learners" into a single strong learner in an iterative fashion. It is easiest to explain in the least-squares regression setting, where the goal is to "teach" a model F to predict values of the form y ^ = F (x) by minimizing the mean squared error.

5. Screenshots

5.1 Accuracy Score

```
Accuracy Score of Linear regression on train set 11.829233894211866
Accuracy Score of Decision Tree on train set 100.0
Accuracy Score of Random Forests on train set 94.20011054244306
Accuracy Score of Gradient Boosting on train set 65.49517152859553
Accuracy Score of Linear regression on test set 36.82102872436395
Accuracy Score of Decision Tree on test set 57.74375293458696
Accuracy Score of Random Forests on test set 74.75284787374565
Accuracy Score of Gradient Boosting on testset 72.43849411693184
```

Fig 5.1 Accuracy Score

Above Figure shows the output of R2Score so we get highest accuracy in Random forest algorithm i.e. 94.20 on Training Data set and 74.75 on Testing Dataset

6. Conclusion

The ML algorithm that perform the best was Decision Tree Model with RMSE = 2680 which got me in the first 42%. The next step will be looking at hyper parameter Tuning and Ensembling.

We have implemented an algorithm that allows us to predict the amount of the basket of a consumer knowing his gender, marital status, age, occupation and number of years spent in the current city of residence. This algorithm is based on easily identifiable low-level indicators and can be used to estimate the purchasing potential of a consumer. In the visualization part, we were also able to draw the profile of the perfect consumer to target in priority marketing campaigns including (Black Friday or others). A targeted or retrospective analysis that takes into account all the indicators of the dataset used could complement this study and determine in particular the products generating the most sales.

7. Bibliography

- Barbaro, M. (2006, November 25). Attention, holiday shoppers: We have fisticuffs in aisle 2 [Electronic version]. The New York Times Late Edition. Retrieved June 11, 2007 from LexisNexis Academic database.
- Bellizzi, J. & Hite, R. (1992). Environment color, customer feelings, and purchase likelihood, Psychology and Marketing, 9, 347-363.
- Bitner, M. (1992) Servicescapes: The impact of physical surroundings on customers and employees. Journal of Marketing, 56, 57-71.
- Black Friday (n.d.). About.com: retail industry located at http://retailindustry.about.com/od/abouttheretailindustry/g/black_friday.htm
- Cline, R. (2005, December 29). Shopping frenzy strikes one day after Thanksgiving. University Wire. Retrieved June 23, 2007 from LexisNexis Academic database.
- Dollard, J., Miller, N., Doob, L., Mowrer, O., & Sears, R. (1939). Frustration and aggression. New Haven: Yale University Press.