Black Friday

Domain Background

Black Friday is the Friday after the thanksgiving. This day is kind of the day where the shopping season starts. In that day all stores come out with very good sales in order to attract most customers and to also clean their warehouses as if the products stay longer they may result in profit loss. The trend lately tends to make those Friday sales not just after thanks giving but many times per year. That is why most companies try to learn their customers behaviour in order to make the best sales that satisfy customer needs and make company gain maximum profit. A problem that is similar to this one is provided on kaggle https://www.kaggle.com/c/ga-customer-revenue-prediction , where the person is challenged to analyse Google merchandise also called store and predict revenue per customer as they will use the results to make actionable operational changes and make a better use of marketing budgets.

Problem Statement

The problem at hand is called Black Friday, the problem revolves around a company called ABC Private the company want to understand and analyse the behaviour of its customers in order to have the best sales plan and promotional strategies possible. They decided that the best way for that is to make a model that predicts the amounts that the customers will buy from the shop products and how much from each category. By having a good time series model their sales plans and promotional strategies will have the highest efficiency possible. Many big companies use such systems to make their sales plans and promotional strategies such as amazon or google where it posted recently a competition on kaggle requesting the contesters to make a model that predicts customers revenue. According to Google this also helps in determining the marketing budgets.

The goal of this model is to be able to predict the amount of purchase a person will buy based on some special features he has. The dataset comes from http://analyticsvidhya.com the dataset contains 550,000 rows of customers, the attributes provided are (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), The dataset overall is not biased which is great because that will make model not biased and increase accuracy. The way I approached this problem is by transforming the Purchase”What needs to be predicted” to fit normal distribution, that helps the model a lot in increasing accuracy of model and make it more predictable. After wards I filled any missing data in the attributes. I removed some of the attributes as they might have low or negative effect on my model such as occupation due to its diversity in the number of occupations and diversity of salary within each occupation. Then I used linear regression and Naive-Bayes as benchmark models, i used XGBRegressor and RandomForestRegressor as my main models.

Evaluation Metrics

The performance will be evaluated on the basis of my prediction of the purchase amount by doing K-Fold cross validation. The metrics I will be using to evaluate the accuracy of my model will be RMSE ("Root Mean Squared Error") . The reason why I will use RMSE is because RMSE is a suitable general-purpose error metric. Compared to the Mean Absolute Error, RMSE punishes large errors.Another benefit of using RMSE over something like MSE is that the error is in the same unit as the prediction. So i can quickly determine how far off my prediction is.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(y_{true\_i} - y_{predicted\_i})\wedge 2}{n}}$$

- n = number of rows
- y_{true_i} = The actual number of the predicted row
- $y_{predicted\_i}$ = The predicted number for the row

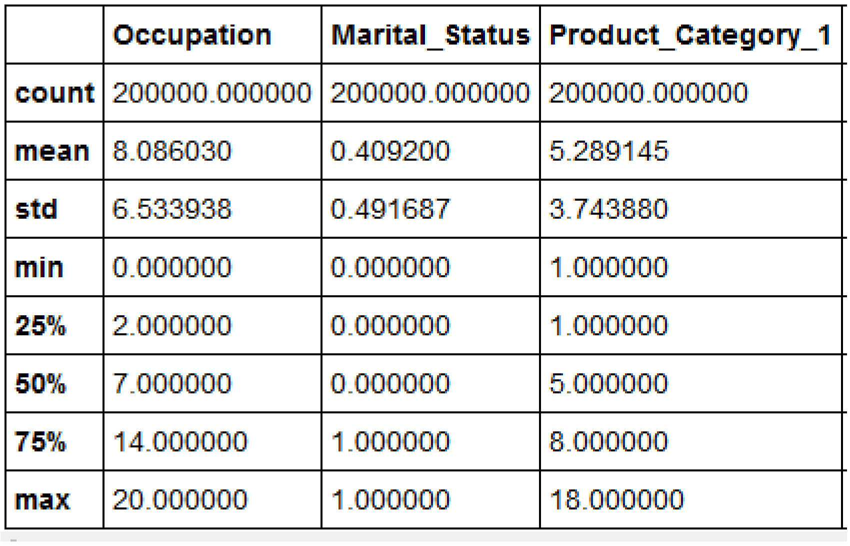
The equation sums the square difference between the the actual value and the predicted value for each row, then it divides the sum of the square difference over their count, after we get the result we get the square root of it.

Data Exploration

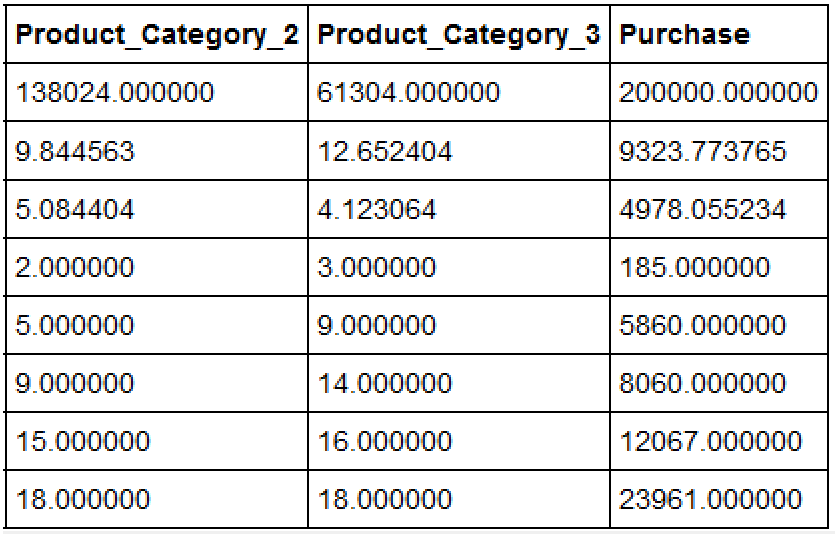
To Begin with the analysis I used 200,000 rows out of total dataset to save time and I saw that having any more rows than that will not mount to much when training the model. I started with describe function in pandas in order to know the minimum, maximum, mean and the quantiles also to know which attributes are numerical and which are categorical. Then I used describe function with categorical values to know how many unique values each category has and what is the frequency of the most repeated value in categorical attribute, also to know if the data frame has very high bias or not because that would affect my accuracy and make it unrealistic. Then checked the sum of null value in the data frame to see if there are any missing values i need to deal with, the only null values were in Product_Category_2 and Product_Category_3 which made sense since not all products fall within one category.

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
Product_Category_3	Product may belongs to other category also
Purchase	Purchase Amount (Target Variable)

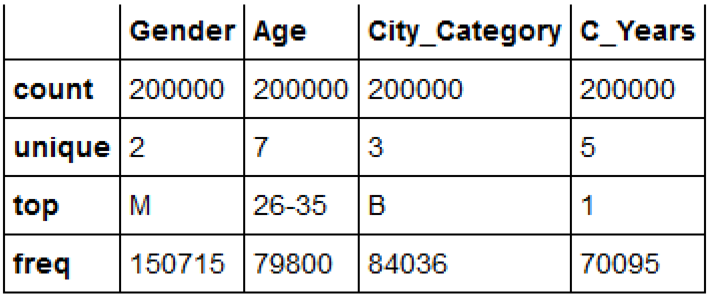
User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
1000001	P00069042	F	0-17	10	A	2	0	3			8370
1000001	P00248942	F	0-17	10	A	2	0	1	6	14	15200
1000001	P00087842	F	0-17	10	A	2	0	12			1422
1000001	P00085442	F	0-17	10	A	2	0	12	14		1057
1000002	P00285442	M	55+	16	C	4+	0	8			7969
1000003	P00193542	M	26-35	15	A	3	0	1	2		15227
1000004	P00184942	M	46-50	7	B	2	1	1	8	17	19215
1000004	P00346142	M	46-50	7	B	2	1	1	15		15854
1000004	P0097242	M	46-50	7	B	2	1	1	16		15686



	Occupation	Marital_Status	Product_Category_1
count	200000.000000	200000.000000	200000.000000
mean	8.086030	0.409200	5.289145
std	6.533938	0.491687	3.743880
min	0.000000	0.000000	1.000000
25%	2.000000	0.000000	1.000000
50%	7.000000	0.000000	5.000000
75%	14.000000	1.000000	8.000000
max	20.000000	1.000000	18.000000



Product_Category_2	Product_Category_3	Purchase
138024.000000	61304.000000	200000.000000
9.844563	12.652404	9323.773765
5.084404	4.123064	4978.055234
2.000000	3.000000	185.000000
5.000000	9.000000	5860.000000
9.000000	14.000000	8060.000000
15.000000	16.000000	12067.000000
18.000000	18.000000	23961.000000



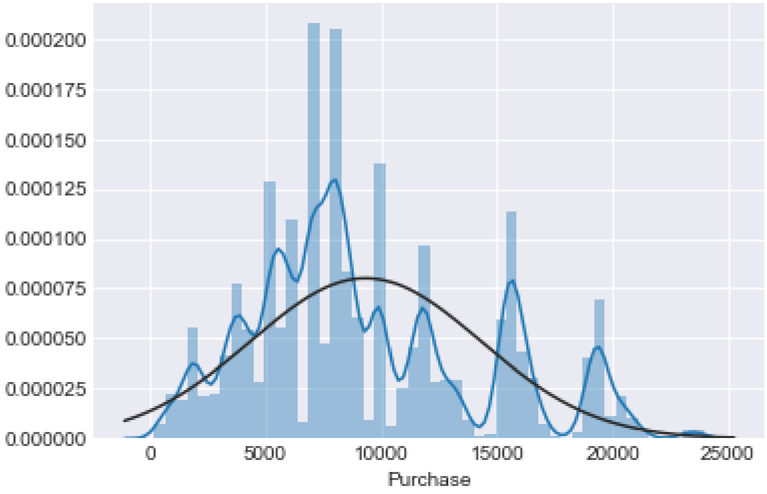
	Gender	Age	City_Category	C_Years
count	200000	200000	200000	200000
unique	2	7	3	5
top	M	26-35	B	1
freq	150715	79800	84036	70095

The occupation ranges between 0-20 where each real number presents an occupation the graph shows that most of the people that came has an occupation that varies between 0-8,the marital status shows is either 0 or 1, 0 = single and 1 = married, The median is equal 0 which is reasonable as all age categories can be married except the age category 0-17 where they are children and they can't get married yet by that age. Product_Categories_1 range from 1-18 where each real number is a product category.

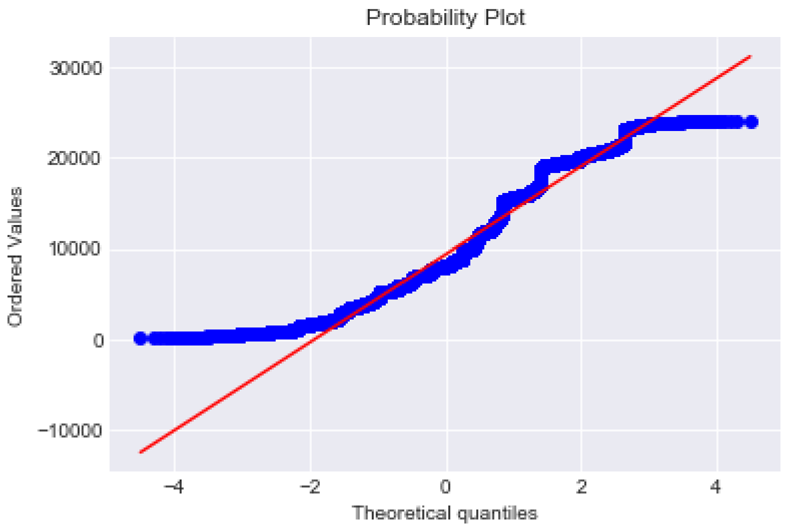
Product_Categories_2 range from 2-18 where each real number is a product category, it has missing values which is reasonable as not each product fall within more than 1 category.Product_Categories_2 range from 2-18 where each real number is a product category, it has a lot of missing values which is reasonable products rarely fall within more than 2 categories. Gender is either M or F it's weird that the frequency of males in dataset is high, 75% of total dataset sample, Age is divided into 7 unique values (0-17, 18-25, 26-35, 36-45, 46-50, 51-55, 55+), it's normal that almost 40% of the buyers are between 26-35 as at this age people start to get married and get a sable job and most of them start to get good salaries. City_Category is divided in (A, B, C). Stay_in_current_city_years(C_Years) ranges between 0-4+, its normal that the most frequency is 1 as by staying in a city for 1 year a person gets used to the city and gets start to settle down and need new items in his house.

Exploratory Visualization

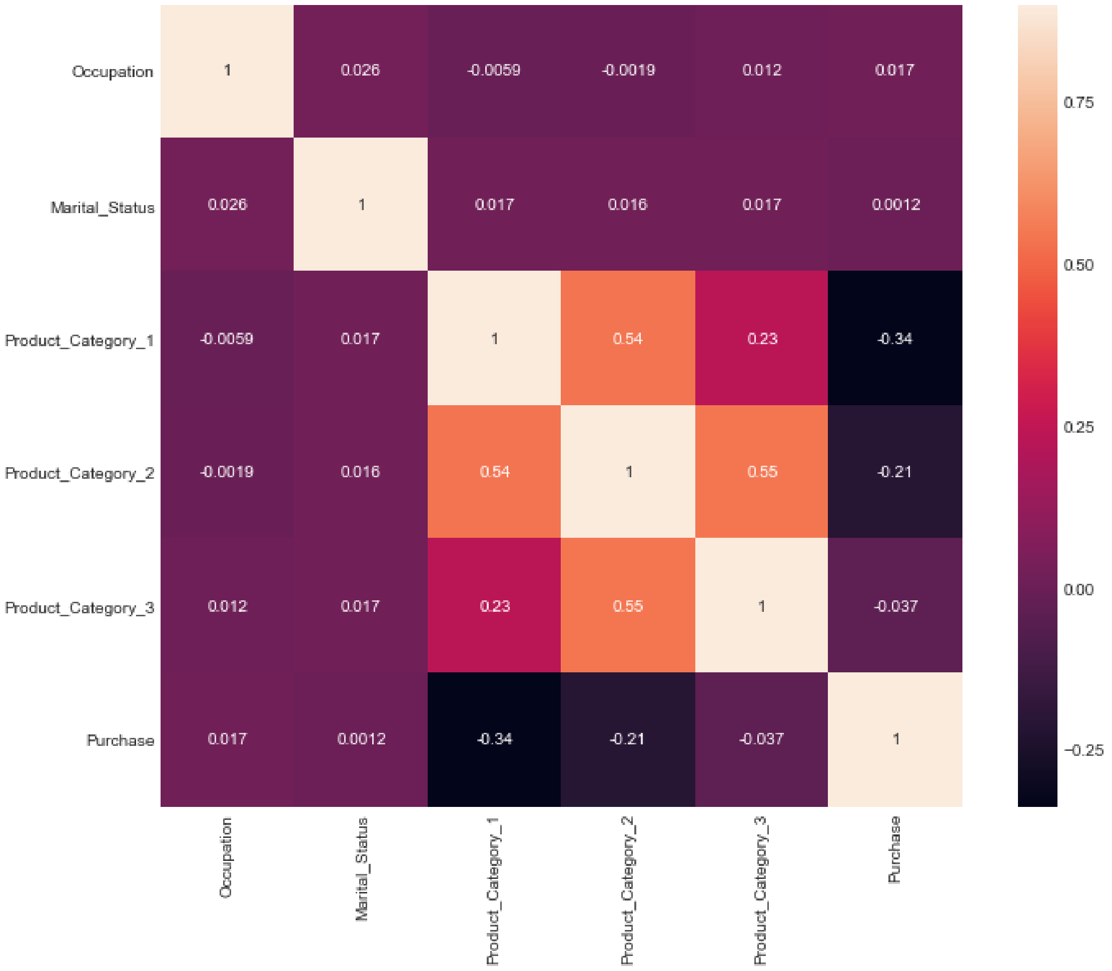
I made a distribution plot on purchase to see how are the values distributed and compare it to normal distribution, The data was very inconsistent, for example the 2 bars that had the highest frequency had between them a bar with very much lower frequency, also after the frequency drops to almost 0 before 15000 and 20000 frequency starts to significantly rise again.



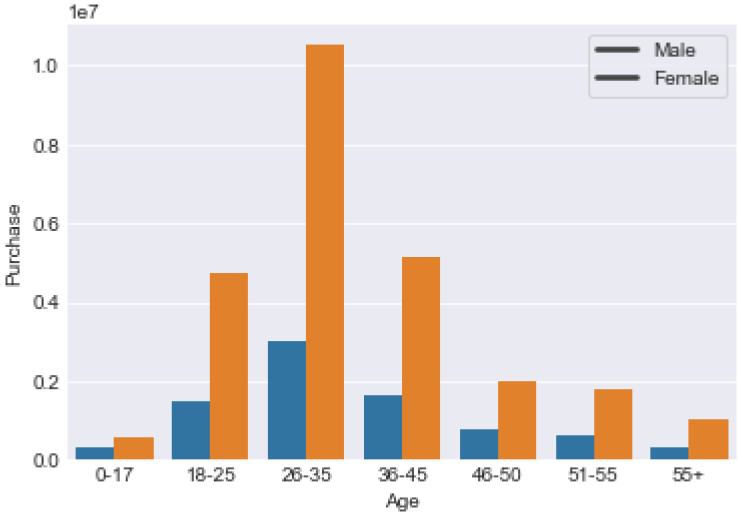
I used probability plot afterwards to have clearer view of purchase data and see if it needed any transformation or not and if it does what type of transformation is required.



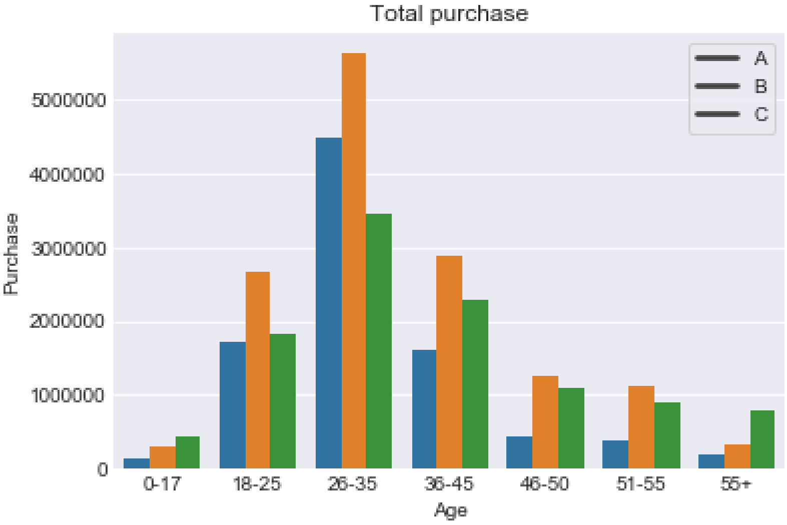
After that I made a correlation graph in order to grasp how much each attribute is correlated with each other and more importantly how is it correlated with Purchase in the data frame. I started making bivariate analysis to see how are the attributes related with each other.



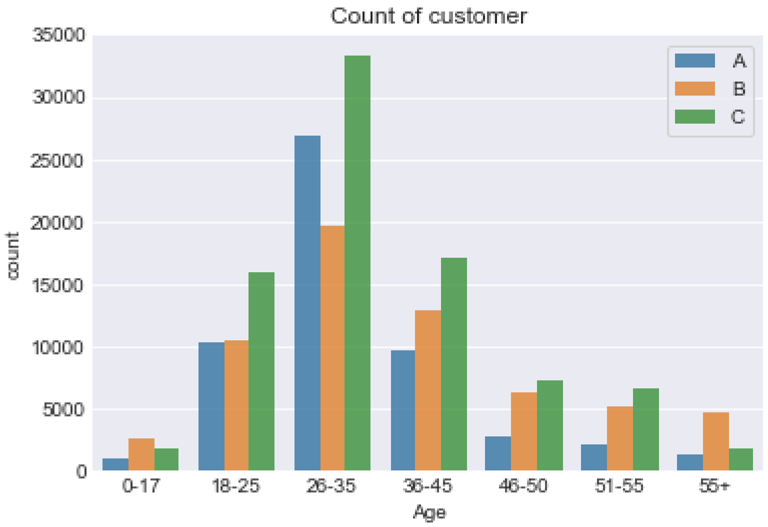
I started with seeing how much are the males and females purchase amount within certain age range, turns out females always buy more than males no matter what is the age and the age both males and females buy the most is in between 26-35 also how much males buy between age 18-25 and 36-45 are very close which is a bit of a surprise that also applies to females, between age 0-17 purchase amount is low and the gap of purchase between males and females is not big like the other age gaps which makes sense since they are still young. After age 45 the amount of purchase start dropping each year passes by.



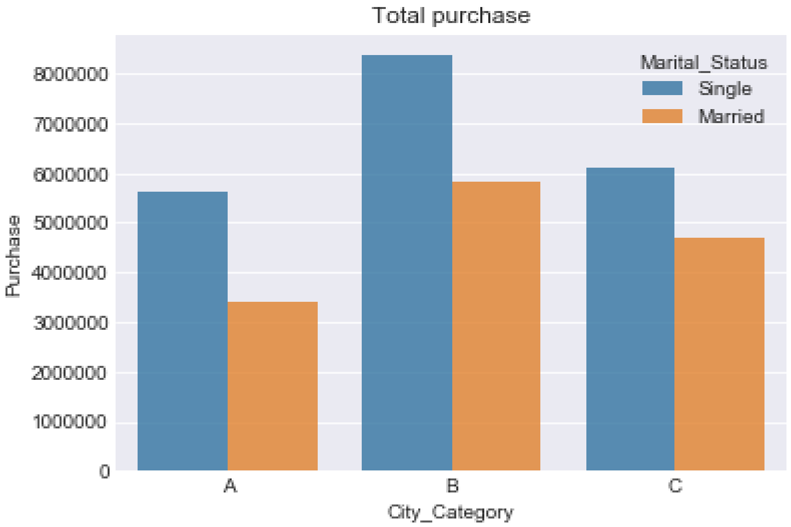
Then after that I compared the amount of purchase with the City_Category and age to determine whether a certain city is favourable within certain age gap or basically a city of them has more population than others or shop is bigger than the others. In the graph city B had the most total purchase in all years except between 0-17 and 55+ which indicates that City_Category B was probably in a large city.



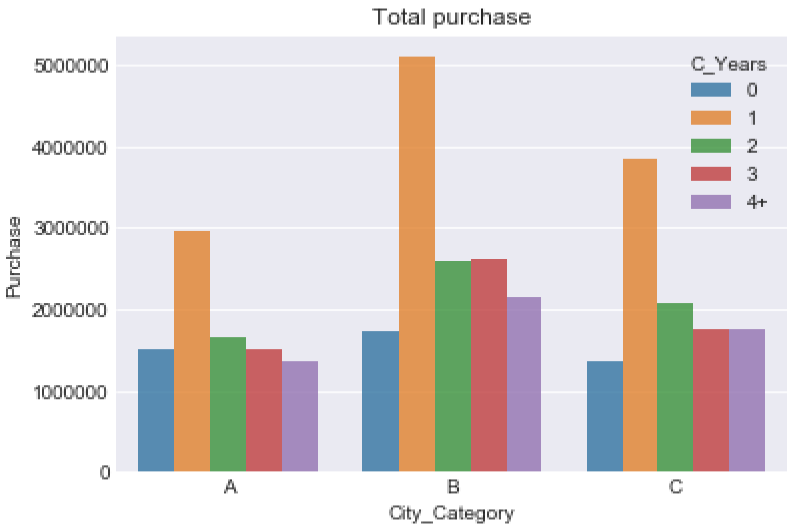
I did a count plot to see how the amount of customers that visit each City_Category and their ages, to my surprise City_Category C had more customers than City_Category B in all age gaps which means that City_Category B is not in a more populated city but the city that the shop in is wealthier that the other City_Categories so people tend to buy more on this city.



I made a comparison to see if single or married people bought more in each city category, turns out that single people buy more than married people regardless of city which indicates that single people have more money to spend because they don't have to spend money on children tuition or the extra bills that comes along side living beside some one else at home.



Tried to compare amount of purchase in relation to amount of time client stayed at city and the city category, turns out that in all cities the customers that stayed exactly one year at the city are the customers that buy things the most, year 0, 2, 3, 4, 5+ were relatively the same in all the cities, which was quite expected as one year duration is the duration where the customer starts settling down in the city and starts to have stable income and stable residence and starts thinking about getting new products to fill his house with.



I tried to see how which product the customers prefer in relation with their age so i counted the amount of products in relation with their age, as guessed the produced data frame was super big to graph so i printed the data frame instead, turns out product 1,5 and 8 are bought in big numbers regardless of the age. Last thing I did was to compare the mean of purchases against the values in Age, Gender, City_Category and C_Years to my surprise the means in all of them were almost the same with maximum 20$ standard deviation.

Algorithms and Techniques

First I imported Scipy in order to use stats library from that library I used boxcox and skew it to make data follow the uniform distribution. I imported KFold and cross_val_score from sklearn.cross_validation to do K-Fold cross validation on the benchmark and my model, also to use it to draw the learning curve and get the mean score from the k tries. From sklearn.naive_bayes i imported GaussianNB in order to use this algorithm along with from LinearRegression that I imported from sklearn.linear_model as a benchmark model. From sklearn.ensemble I imported RandomForestRegressor and from xgboost I imported XGBRegressor, I imported them to use as my main model. From sklearn.model_selection I imported GridSearchCV to use it to search for the best parameters that helps in having the best score. From sklearn.model_selection i imported learning_curve to use to show the how well model is learning and show the accuracy in simple way.

The GaussianNB uses something called Bayes theorem it is a theorem that works on conditional probability,conditional probability is calculating the probability and event will happen based on previous events.

$$P(H \mid E) = \frac{P(E \mid H) * P(H)}{P(E)}$$

where

- P(H) is the probability of hypothesis H being true. This is known as the prior probability.
- P(E) is the probability of the evidence(regardless of the hypothesis).
- P(E|H) is the probability of the evidence given that hypothesis is true.
- P(H|E) is the probability of the hypothesis given that the evidence is there.

(“Copied from http://dataaspirant.com/2017/02/06/naive-bayes-classifier-machine-learning/ ”).

The linear regression algorithm works by trying to adjust the slope line to find the best line that predicts the value Y from the one or more attributes “X”, the way it does they is buy finding the minimum square sum between points above and below the line.The XGBoost library implements the gradient boosting decision tree algorithm, it produces a model which was the result of several models before it, even time it trains model it learns from the model that is before it, most of time these models are decision trees. RandomForest makes multiple Decision tree models at the same time. Then choose the mode of the classes this helps in avoiding over fitting or under fitting in the model.

Benchmark model

The benchmark model I used was Naive-Bayes and linear regression. The reason I used them is because they are simple and computationally very cheap compared to other algorithms. This Naive-Bayes score is usually low if compared with other model but the score is relatively reasonable and can be accepted as benchmark,The reason I used Naive Bayes as benchmark is because it depends only on one attribute to predict the model, while in linear regression it uses a linear line in order to predict all models. In naive-bayes using only one attribute to predict produces low accuracy while in linear regression the model lack flexibility. Linear regression fits is also good as a benchmark. I got the score of both of them by doing K-Fold cross validation on both of them. The scoring unit I used is RMSE. The score I got for Naive-Bayes was 49.81 while the score for linear regression was 44.95.

Data Preprocessing

First thing I started with is that I imported only 200,000 out of 500,000 rows as I deemed saw that they are enough to model a very good model and any more rows than that will not have much impact on my model, also to save time since I don't use GPU. I dropped userid since I don't need it and it doesn't has any use in my model and also dropped Product_id since I using product id to classify would cause a lot of problems and would be hard to get any sort of information out of it, for example there are tons of products that fall under food category but one with their unique id also new product id's may come during testing since the amount of product id's is too big to count. I started looking at the distribution of data and saw that it somehow follows normal distribution but it didn't fit the distribution nicely so I imported a class called boxcox this call gives me a value lambda based on the distribution I want the model to fit in, Then I applied that transformation value on Purchase attribute which made it much more uniform and consistent. After that I changed the values of Stay_In_Current_City_Years to become 0 if it is [0, 2, 3, 4+] and 1 if it is one year because the analysis showed that there is a spike in the amount of purchases someone buys if he has stayed exactly one year at the city he is in.I decided to get dummies of the Gender is it has only two categorical values in it and as the analysis has shown females buy more than males no matter the age, I used dummies instead of LabelEncoder so that the males can also have some sort of value. I also decided to use dummies with City_Category since analysis showed that some City_Category may be favourable within certain age more than other City_Categories. I decided to drop occupation since I saw that it has very low correlation with purchase, there are many occupations to even count and when testing a new occupation might come up then the model will now know how to handle it and lower the accuracy, also the occupation is not a good indicator on how much money a person owns as a person can make double the money a person does with same occupation. I dropped the Product_Category_3 attribute due to the fact that almost 80% of the rows in this attribute are NAN values, I will replace the NAN values that are not 0 since the NAN value might be used to indicate something so I want to give them some weight. I think Product_Category_1 and Product_Category_2 are enough to identify the class the product. I used fit transform on Age attribute so I can use it in my model, I didn't use get dummies as Age has 7 unique values and when training my model having so much attributes might cause my model to fall in the curse of dimensionality, this will make my model take much more time to train and might cause my accuracy to fall. In the end I checked the minimum value in Product_Category_2, it turned out to be 2 which was perfect for me, i substituted the NAN values in Product_Category_2 with one instead of 0 as i wanted to give NAN values some sort of weight as they can be used to indicate that a product is specialized only in a certain category which will help in classifying later on.

Implementation

At first I specified a The first two algorithms I used were Naive-Bayes and Linear regression, I used them at first as they were my benchmark model. In Naive-Bayes the I used the default parameters which were "*priors=None, var_smoothing=1e-09*" and with Linear Regression I used "*fit_intercept=True, normalize=False, copy_X=True, n_jobs=None*". I validated their score by using cross_val_score function where KFold parameters are “(len(y), n_folds=10, shuffle=False, random_state=0)” and the scoring function I used was 'neg_mean_squared_error' then I removed the negative from the score and used sqrt function on it to get the RMSE. I specified two algorithms to use as my main model and those are XGBRegressor and RandomForestRegressor. For XGBRegressor I used the parameters “(base_score=0.5, booster='gbtree', colsample_bylevel=1, colsample_bytree=1, gamma=0, learning_rate=0.15, max_delta_step=0, max_depth=6, min_child_weight=1, missing=None, n_estimators=100, n_jobs=1, nthread=None, objective='reg:linear', random_state=0,

reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None, silent=True, subsample=1)”.

While for RandomForestRegressor I used the parameters “(bootstrap=True, criterion='mse', max_depth=8,max_features='auto', max_leaf_nodes=None,min_impurity_decrease=0.0,min_impurity_split=None, min_samples_leaf=1, min_samples_split=2,min_weight_fraction_leaf=0.0, n_estimators=60, n_jobs=1, oob_score=False, random_state=None, verbose=0, warm_start=False)”.

I validated their score by using cross_val_score function using KFold and the scoring function I used was 'neg_mean_squared_error' then I removed the negative from the score and used sqrt function on it to get the RMSE.

Refinement

To do refinement I used the function GridSearchCV I used it in order to find the best parameters available in my model, For XGBoostRegressor the parameters I used GridSearchCV on were 'booster' with values ('gbtree','gblinear'), 'max_depth' with values (2,3,4,5,6,7,8) and 'learning_rate' with values (0.25,0.2,0.15,0.1,0.05). The best values chose for this algorithm were 'booster'='gbtree', 'max_depth'=6 and and 'learning_rate'=0.15. For RandomForestClassifier the parameters I used GridSearchCV on were 'n_estimators'with values (60,80,100,120,140) and 'max_depth' with values (6,7,8). The best values chose for this algorithm were 'n_estimators'=60 and max_depth=8. I kept changing the values in the parameters a lot between the 2 algorithms to make sure it choose the parameter closest to best solution as much as I can.

Model Evaluation and Validation

I used XGBRegressor the reason I used it is because It is a highly flexible and versatile tool that can work through most regression, classification and ranking problems as well as user built objective functions. As an open source software, it is easily accessible and it may be used through different platforms and interfaces.This model and its parameters are chosen based on many tries by using GridSearchCV,The final score is considered to be fairly good as an RMSE score of 28.37 with the purchase attribute having the mean value of 169 and Standard Deviation of 47.

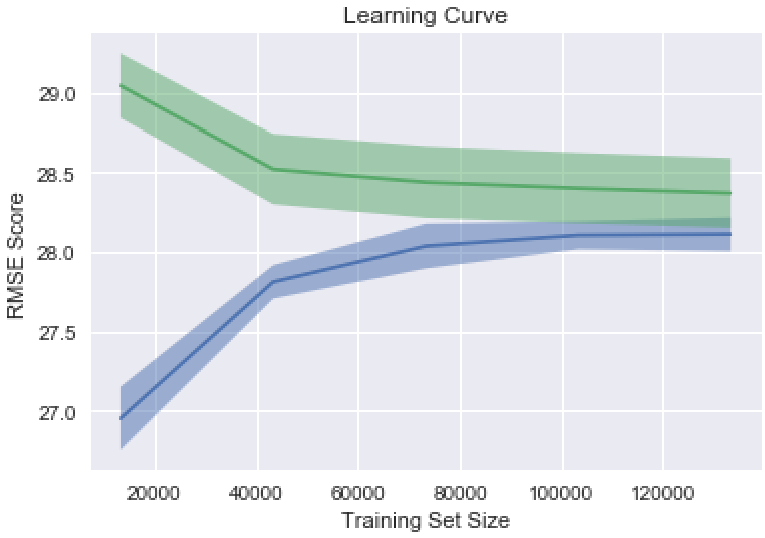
The reason I used RandomForestRegressor is It is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier.It runs efficiently on large databases.It can handle thousands of input variables without variable deletion.It gives estimates of what variables are important in the classification.It generates an internal unbiased estimate of the generalization error as the forest building progresses.This model and its parameters are chosen based on many tries by using GridSearchCV,The final score is considered to be fairly good as an RMSE score of 28.50 with the Purchase attribute having the mean value of 169 and Standard Deviation of 47. This model can be trusted as the model has been trained by doing 10 KFold cross validation with shuffle = False on 200,000 rows so if there was a unique value in them it would appear on the model score.

Justification

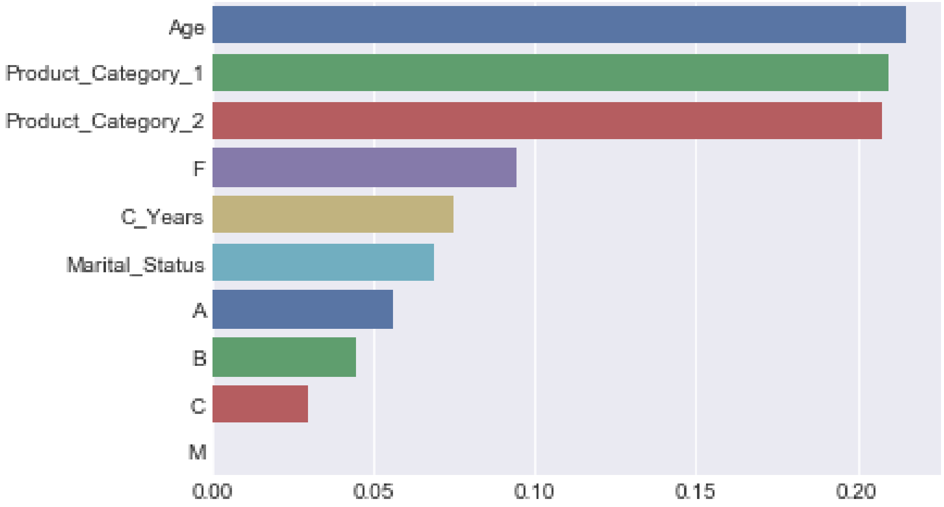
The Score of both models are better than the benchmark models by at least at least 17 RMSE units that means that my model is better by 37% than my benchmark model . This score of my models have very good RMSE score considering the margin of error.

Free-Form Visualization

At first after I chose the model with the better score I made a learning curve using the best parameters that GridSearchCV selected, then I plotted the curve to see how well model model is learning.



After it I used feature importance to know the weight of each attribute in training my model and the order of importance came as follows (Age, Product_Category_1, Product_Category_2, F, C_Years, Marital_Status, A, B, C, M).



Reflection

Every aspect has been documented, what was interesting is that while I was trying different parameters for the models the order of importance of feature in feature selection change for example in XGBClassifier the importance of first 3 attributes were (Product_Category_1, Product_Category_2, Age) when I used parameters max_depth = 5 and learning_rate = 0.2, but the order changed as I changed the parameters, even though the change was small the effect of it on feature importance was big. Training the models is definitely the most hard part, the amount of time alone spent in training all the models took very long time, it is one of the main reasons I didn't use Support Vector Regression as i ran the model and it took more than 7+ hours of running time and the training the model didn't end so I had to terminate the run. The model I chose fit my expectations as the RMSE score is 28 while the Standard Deviation of the data is 47 which means my error margin within 60% of Standard Deviation.

Improvement

The model can be improved a lot If I used the whole 550,000 records in the dataset but I couldn't because I lack the computing power to handle this amount. Every algorithm I don't know how to implement I research it till I can solve it. Of course there is always a better solution than the one you currently have, there are all sorts of algorithms that I don't know about them completely so that's why I didn't chose them but they are better fit for the problem than the algorithm in chose, Also I believe the parameters of the algorithm can be tuned even further to have much better accuracy.