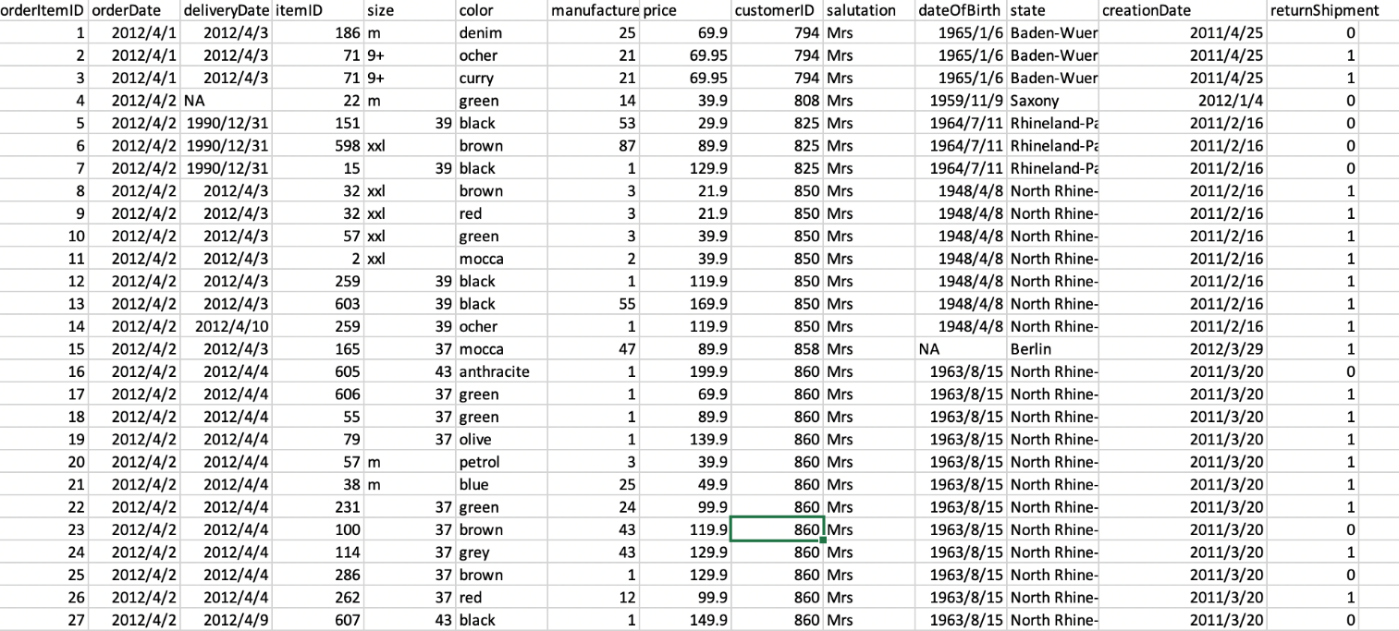
**RETURN PREDICTIONS**

1. **Introduction :**

Return constitute a very important part of cost in online selling. Therefore, the prediction of the return can help a lot. In order to make the prediction, we use the historical purchase data of an online shop to predict a certain purchase is converted into a return or not. The return “yes/no” is our target variable. The business case is that we need to predict the probability of returns that result in returns (1 in this case) so that the online retailer is able to work on parameters like restriction on the basis of payment option, shipping costs, sizing guides etc. So our aim with the data in hand was to predict the maximum amount of return cases.

1. **Data overview：**



In our data set returnShipment is the target variable where the value 1 means the item was returned while 0 means item was not returned. In the data there are some missing values, string values, dates and some really weird values which we have tried to work on through our feature engineering.

1. **Feature engineering**

**3.a Column: deliverytime\_days**

We are interested in whether the deliverytime will affect the return or not, so we use two column “ delivery date” and “order date” to create the new column “deliverytime” to see whether the duration of delivery will influence the return or not. Here delivery time is equal to order date subtracted from the delivery date. When we see the plot for the delivery time some of the time is in negative which is not possible. We have removed the negative delivery time from our data set as these are the outliers.

**3.b Column: age\_at\_order\_yrs**

We create a column called “age\_at\_order\_yrs” to understand about the age of the consumer during the time of ordering and to see the influence of the age of the customer on return. This column created by subtracting “dateofBirth” from “orderdate” to see how old the customer. We have again removed all the ages which are negative as it is not possible and is an outlier.

**3.c Column: product\_age\_days**

When the column “creationdate” is subtracted from the “orderdate” , we can see a new column “product\_age\_days”

**3.d Column: Order Date**

We also took out the day of the week when the product was ordered to understand if that effects our returns or not

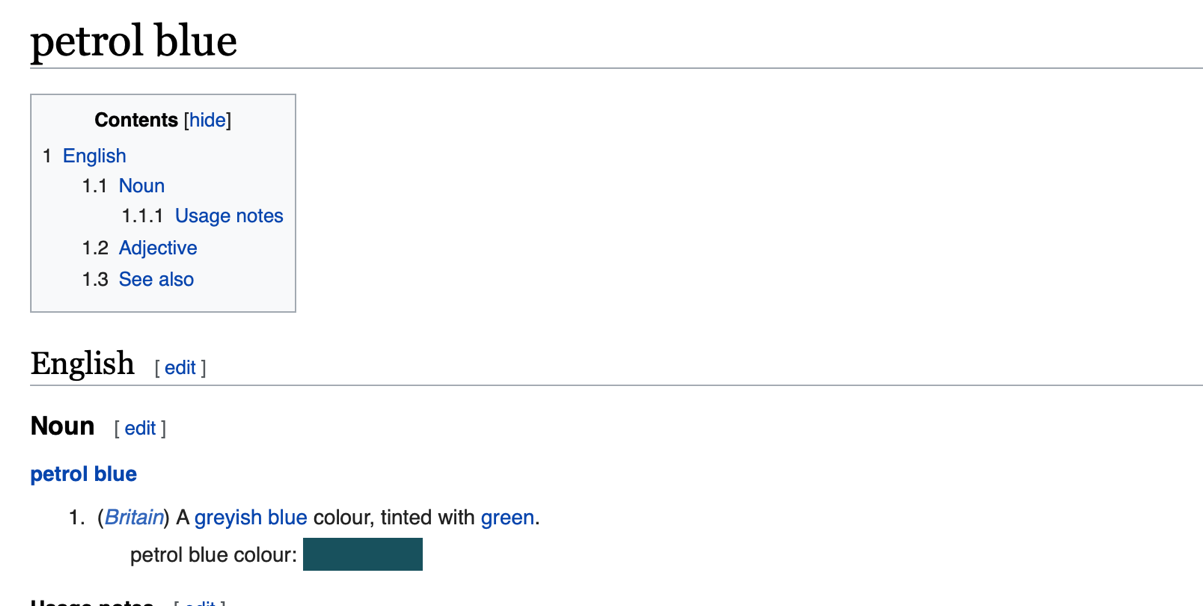
In the end, for the column” **deliverytime\_days”,” age\_at\_order\_yrs” and” product\_age\_days”,** because they are dates and we have extracted all the valuable columns from them.

﻿**3.e Column: size of the item**

This column is difficult to understand as it has mix of varied data in different representation. The item here can mean anything from shoes, clothes to pants etc . and the sizes of these articles vary a lot. The size is varied but as 90% of our customers are female so for to convert Mode Merr system( s,m,l,xl,xxl,xxl) into European system of sizes for females,and hence we have removed the + sign and have kept the size so that there is uniformity.

**3.f Column: color**

This column is also complex one. Our data has 87 kinds of color, which are varied so some of them are repeated a lot of times while others are very few. Therefore, we decide to simplified from 87 color to only 12 categories of color. We decided to use the color that has comparatively more item as a category. Mostly these colors were VIBGYOR, white, black and grey. These 12 categories were covering 70% of our data set and rest of the other colors, by the definition of wiki, can be defined to the exist color categories. After this process, we set up the 12 categories of color: which are red, blue, grey, red, brown, green, white, purple, yellow, pink, orange and other.



﻿

**Dummy variables**

We used label binarizer for salutation, color and state as they are not order based and had categorical type of data. We modified our model by adding new variable called day of week and used dummy for them. For color variables, we made assumptions that most of the human beings classify color to the groups, thus we grouped 87 color variability to the main color categories. Eventually, in our model we have 12 dummy variables for color, 15 dummy variables for states, 5 dummy variables for salutation and 7 dummy variables for day of a week. To avoid dummy variable trap we dropped one dummy variable from each dummy group.

For instance, dummy variables for color

**Dummy variables for color**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | black | blue | brown | green | grey | orange | pink | purple | red | white | yellow |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 5 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Dummy variables for salutation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Family | Mr | Mrs | not reported |
| 0 | **0** | **0** | **1** | **0** |
| 1 | **0** | **0** | **1** | **0** |
| 2 | **0** | **0** | **1** | **0** |
| 3 | **0** | **0** | **1** | **0** |
| 4 | **0** | **0** | **1** | **0** |
| 5 | **0** | **0** | **1** | **0** |
| 6 | **0** | **0** | **1** | **0** |
| 7 | **0** | **0** | **1** | **0** |
| 8 | **0** | **0** | **1** | **0** |
| 9 | **0** | **0** | **1** | **0** |

Dummy variables for states

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Bavaria | Berlin | Brandenburg | Bremen | Hamburg | Hesse | Lower Saxony | Mecklenburg-Western Pomerania | North Rhine-Westphalia | Rhineland-Palatinate | Saarland | Saxony | Saxony-Anhalt | Schleswig-Holstein | Thuringia |
| 0 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| 1 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| 2 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| 3 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** |
| 4 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** |
| 5 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** |
| 7 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** |
| 8 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** |
| 9 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** |

Dummy variables for days of week

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Monday | Tuesday | Wednesday | Thursday | Saturday | Sunday |
| 0 | **0** | **0** | **0** | **1** | **0** | **0** |
| 1 | **0** | **0** | **0** | **1** | **0** | **0** |
| 2 | **0** | **0** | **0** | **1** | **0** | **0** |
| 3 | **0** | **1** | **0** | **0** | **0** | **0** |
| 4 | **0** | **1** | **0** | **0** | **0** | **0** |
| 5 | **0** | **1** | **0** | **0** | **0** | **0** |
| 6 | **0** | **1** | **0** | **0** | **0** | **0** |
| 7 | **0** | **1** | **0** | **0** | **0** | **0** |
| 8 | **0** | **1** | **0** | **0** | **0** | **0** |
| 9 | **0** | **1** | **0** | **0** | **0** | **0** |

After creating dataframes of these dummy variables we add it in our data set and remove columns like color, salutation, state and day of the week columns

**Downsampling**

After finishing most of the feature engineering part, our data is quite imbalanced. 60% of data consist from returned data. We took random sample of our data so that our size of sample of returned products are equal to number of unreturned products. After downsampling we have 127489 products in both returned and not returned products.

**1 151917**

**0 127489**

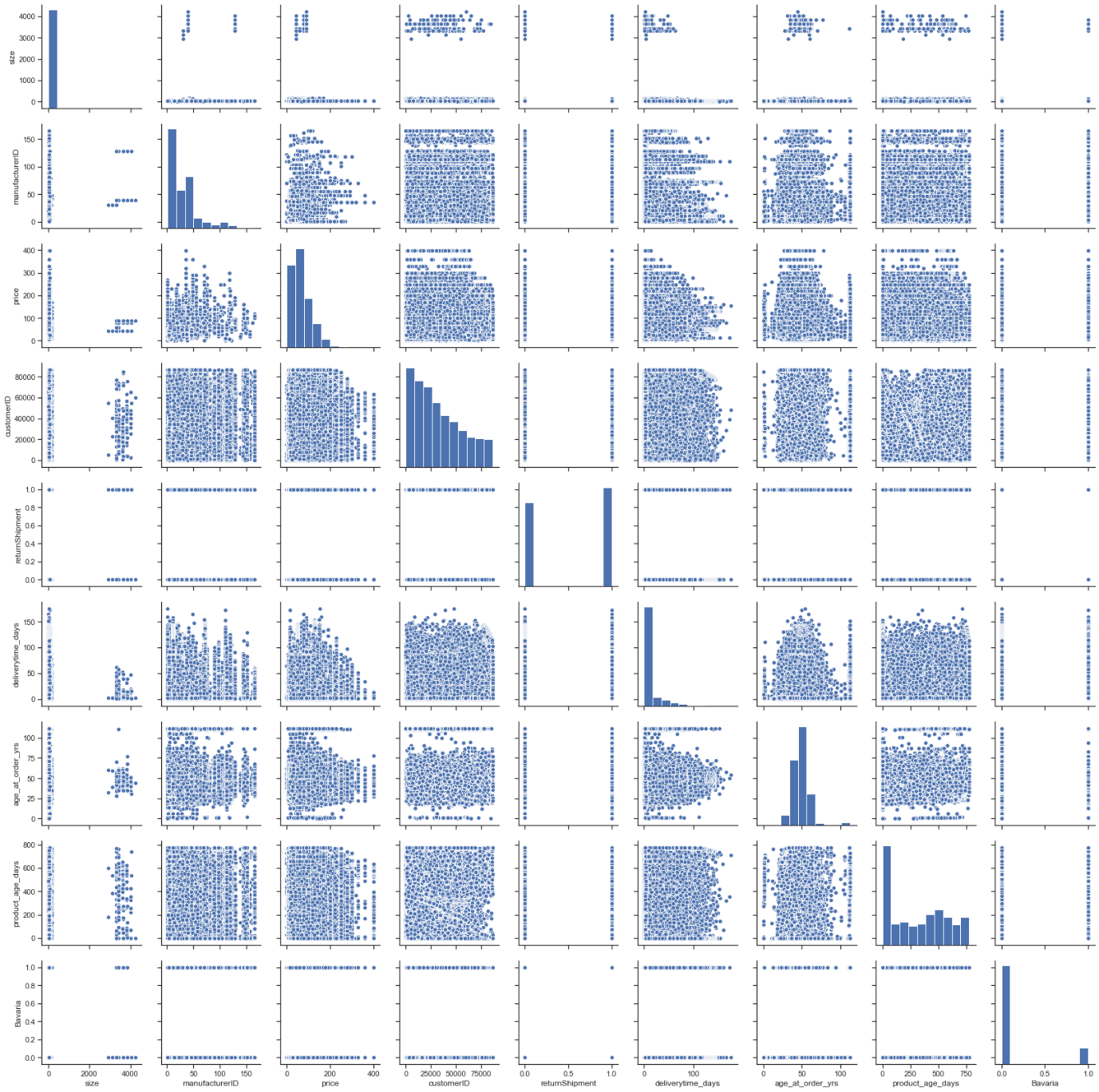
**Name: returnShipment, dtype: int64**

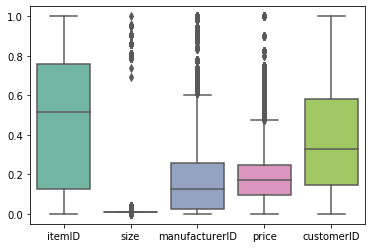
**1 127489**

**0 127489**

**Name: returnShipment, dtype: int64**

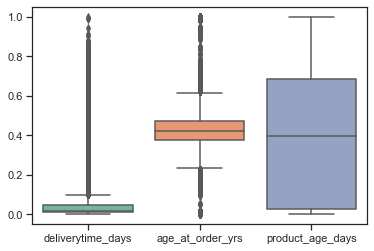
**Data Visualization**

****



Boxplot 1

From 1st boxplot we can that itemID and customerID are normally distributed. However, manufactorID, size and prices are skewed to the left and have a lot of outliers. These outliers in size can be explained by sizes of the jeans as they have four digit size numeration. Regarding the price, boxplot 1 shows that there are few customers that buy expensive products.



Boxplot 2

In Boxplot 2 age of the customer is normally distributed having some outliers in both tails, it means that most of the customers are middle aged persons. Regarding the delivery time we can conclude that delivery policy of this store is quite good and stable.

**Cross Validation**

We used the cross validation to know how our model predicts the outcome of data which it have not seen before. We divided our data into 10 randomly sampled equal parts. Using that method, it results in less biased model comparing to other methods. The best range of k-folds are from 5 to 10.

1. **Modeling**
   1. **Logistic Regression**

We use the logistic Regression model because we only have 2 target variables. This model uses the sigmoid function; if the predicted probability of return is >.5 then the label is 1 and otherwise 0. In the modeling the overall results were not good.

Below is the model accuracy in the case of cross validation (CV = 10). We can see the accuracy is around 56% but is quite similar for both the data sets which means that the model is robust and is not overfitting. The standard deviation is also very low = .003

Accuracies =[0.56321388 0.56566498 0.56399824 0.56880239 0.56483161 0.56801804

0.57572192 0.56822082 0.56449478 0.56748541]

Mean = 0.5670452073428094

SD = 0.0034375585557530714

Model Logistic Regression

Mean Acc. Training 0.567045

Standard Deviation 0.00343756

Acc. Test 0.565848

The accuracy rate of test sample is 56.6%, and the confusion matrix is

Confusion Matrix Testing:

[[15551 9942]

[12198 13305]]

From the chart below we can identify the model also doesn’t give high accuracy for the actual Return (case when output is 1) but has better accuracy for non-return case (case when output is 0)

precision recall f1-score support

0 0.56 0.61 0.58 25493

1 0.57 0.52 0.55 25503

accuracy 0.57 50996

macro avg 0.57 0.57 0.57 50996

weighted avg 0.57 0.57 0.57 50996

* 1. **Naïve base**

We use the naïve base model because we assume that all variable is independent between each other and use the model to calculate the predicted probability of return. In the modeling the overall results were not good.

Below is the model accuracy in the case of cross validation (CV = 10). We can see the accuracy is around 52% but is quite similar for both the data sets which means that the model is robust and is not overfitting. The standard deviation is also very low = .002

Accuracies = [0.52085887 0.52076082 0.51326045 0.52103147 0.52014903 0.51970781

0.51642318 0.51951172 0.51897245 0.51806638]

Model Naive Bayes

Mean Acc. Training 0.518874

Standard Deviation 0.00230353

Acc. Test 0.519198

But what is interesting to see in the naïve base model is the confusion matrix below which clearly states that the model is very good in predicting the case of return (case when output is 1)

Confusion Matrix Testing:

[[ 5785 19785]

[ 4734 20692]]

From the below chart we can analyse that the model has really high accuracy for Return case (where target variable is 1). While really low accuracy for the No-return case (where target variable is 0). But as the business problem is to predict about the return the model can be really useful in our analysis. Though our model is balanced but if we calculate the sensitivity and specificity for it we get

precision recall f1-score support

0 0.55 0.23 0.32 25570

1 0.51 0.81 0.63 25426

accuracy 0.52 50996

macro avg 0.53 0.52 0.47 50996

weighted avg 0.53 0.52 0.47 50996

4.**3 Random Forest**

Random forest is an ensemble classifier that consists of many decision trees and simple majority vote is taken for prediction. The two hyperparameters related with Random forests are the depth of the tree and the random samples. In the first case we checked for the depth of the tree from 4 till 8 and n estimators 10, 50,100,150,200. The best case scenario was for depth 8 and n estimator at 8. Below is the result for it.

{‘max\_depth’: 8.0, ‘n\_estimators’: 200}

Model Random Forest (grid)

Mean Acc. Training 0.583409

Standard Deviation 0.00226173

Acc. Test 0.582203

Name: 2, dtype: object

Confusion Matrix Testing:

[[10676 14661]

[ 6645 19014]]

precision recall f1-score support

0 0.62 0.42 0.50 25337

1 0.56 0.74 0.64 25659

accuracy 0.58 50996

macro avg 0.59 0.58 0.57 50996

weighted avg 0.59 0.58 0.57 50996

The model is robust and is predicting well for return case which is 1.

As the best case scenario was 8 and 200, we checked the algorithm again for depth 8,9 and 10 and with n estimators as 200,210 and 220. The best results were for depth 10 and estimator 200 Below is the result for it

{‘max\_depth’: 10.0, ‘n\_estimators’: 200}

Model Random Forest (grid)

Mean Acc. Training 0.587469

Standard Deviation 0.00201622

Acc. Test 0.584791

Name: 3, dtype: object

Confusion Matrix Testing:

[[11060 14277]

[ 6897 18762]]

precision recall f1-score support

0 0.62 0.44 0.51 25337

1 0.57 0.73 0.64 25659

accuracy 0.58 50996

macro avg 0.59 0.58 0.58 50996

weighted avg 0.59 0.58 0.58 50996

The results were very similar but we again checked in with max depth varying from 11 to 13 and below is the result

{‘max\_depth’: 13.0, ‘n\_estimators’: 240}

Model Random Forest (grid)

Mean Acc. Training 0.593293

Standard Deviation 0.00387035

Acc. Test 0.594949

Name: 0, dtype: object

Confusion Matrix Testing:

[[11808 13628]

[ 7028 18532]]

precision recall f1-score support

0 0.63 0.46 0.53 25436

1 0.58 0.73 0.64 25560

accuracy 0.59 50996

macro avg 0.60 0.59 0.59 50996

weighted avg 0.60 0.59 0.59 50996

As the results were quite similar we didn’t go further. But Random Forest was able to provide good accuracy for the our business case of return (case 1).

* 1. **Decision Tree**

The ID3 model where entropy is used as the measure index. We used cross valida-tion (CV = 10) with measuring index as entropy. Further we kept the depth of the tree from 11 till 17. The results for ID3 were good. With over all accuracy 58% for test and training. Out of all the depth we got the best result for 11. The model is robust as well with this hyperparameter as accuracy of both test and training is similar and is not overfitting. Also the standard deviation is .001 as can be seen below

{'max\_depth': 11.0}

Model Decision Tree (grid)

Mean Acc. Training 0.581365

Standard Deviation 0.00143814

Acc. Test 0.582693

Name: 1, dtype: object

Confusion Matrix Testing:

[[13364 12206]

[ 9075 16351]]

Through confusion matrix we can make out that the model has predicted better for the returns which is our business problem. The below chart clearly highlights all the parameters for 0 and 1.

precision recall f1-score support

0 0.60 0.52 0.56 25570

1 0.57 0.64 0.61 25426

accuracy 0.58 50996

macro avg 0.58 0.58 0.58 50996

weighted avg 0.58 0.58 0.58 50996

**4.5 Gini**

The Cart model where Gini is used as the measure index. We used cross validation (CV = 10) with measuring index as Gini which is by default. Further we kept the depth of the tree from 11 till 17. The results for CART were good and similar to the ID3. With over all accuracy 58% for test and training. Out of all the depth we got the best result for 12. The model is robust as well with this hyperparameter value as accuracy of both test and training is similar and is not overfitting. Alsothe standard deviation is .002 as can be seen below

{'max\_depth': 12.0}

Model Decision Tree (grid)

Mean Acc. Training 0.58112

Standard Deviation 0.00273347

Acc. Test 0.581006

Name: 9, dtype: object

Confusion Matrix Testing:

[[13175 12162]

[ 9205 16454]]

Through confusion matrix as in the case with ID3 we can make out that the model has predicted better for the returns which is our business problem. The below chart clearly highlights all the parameters for 0 and 1.

precision recall f1-score support

0 0.59 0.52 0.55 25337

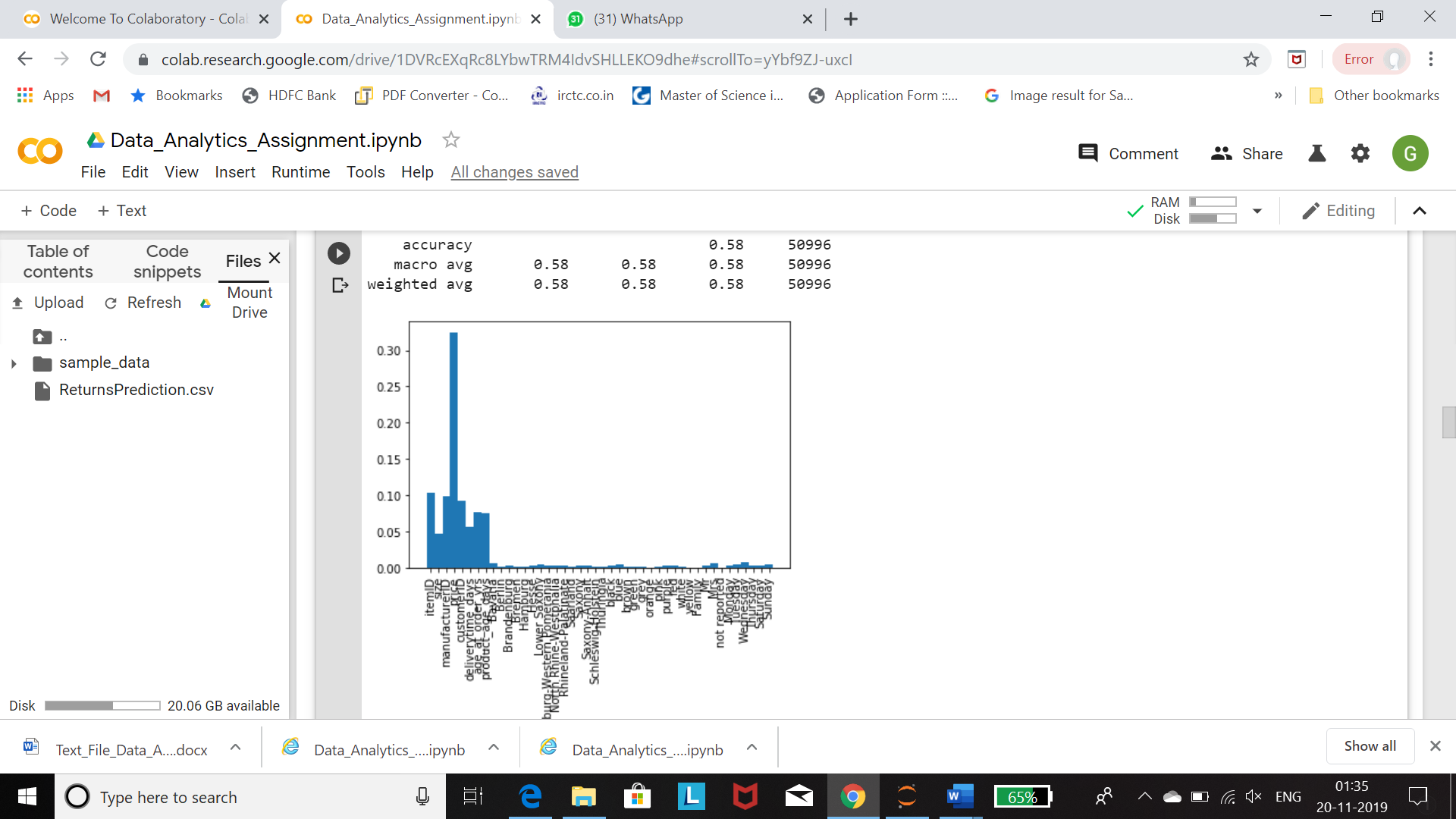
1 0.57 0.64 0.61 25659

accuracy 0.58 50996

macro avg 0.58 0.58 0.58 50996

weighted avg 0.58 0.58 0.58 50996

We also received the chart with the feature importance for Gini which is below



The highest importance is for the price followed by item id and manufacture id. What is interesting to note is that our featured engineer columns delivery time, age of the product and age of the customer have significant contribution as well.

* 1. **Gradient Boosting**

We were not able to compute the cross validation method for the Gradient because our data set is huge and it was taking very long processing time. Through cross validation we wanted to check for the max\_depth of 11., 12., 13.,

subsample for 0.7, 0.8, 0.9, n\_estimators as 200, 220,230 and learning rates 0.1, 0.2, 0.3.

We were able to perform the modelling without cross validation and below were the result. Accuracy for test and training was similar at 60%

Confusion Matrix Training:

[[66017 69701]

[41345 94647]]

Accurray Training: 0.5913069081005484

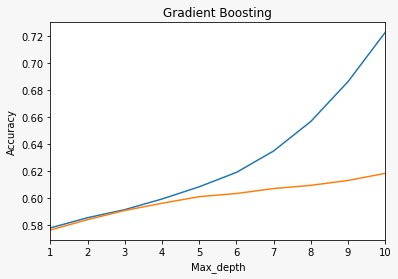
Confusion Matrix Testing:

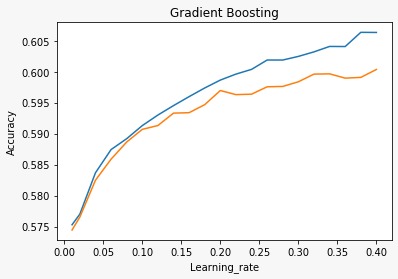
[[16528 17573]

[10230 23597]]

Accurray Test: 0.5906989753857025

Through the below graph we can understand that the best parameters for n depth is 10 and learning rate is .40





* 1. **Discriminant Analysis**

**Linear Discriminant Analysis** (LDA) model searches for best suitable line that separates states of item return. It requires assumptions as normality, independence and multicollinearity. Nevertheless, discriminant analysis is robust to violation of these assumptions which makes it excellent for classifier development.

Accuracy of training and test with 10 k-fold is 56.7% and standard deviation is only 0.003 saying that our model is robust.

Accuracies = [0.56561596 0.56473357 0.57338955 0.56574174 0.56745759 0.57044808 0.56093735 0.56598686 0.56834003 0.56652613]

Model Linear Discriminant Analysis

Mean Acc. Training 0.566918

Standard Deviation 0.00318269

Acc. Test 0.564652

Confusion Matrix Testing:

[[15807 9530]

[12671 12988]]

Through confusion matrix we can identify that model has been able to predict fro the non return case better. Below are the paramaters.

precision recall f1-score support

0 0.56 0.62 0.59 25337

1 0.58 0.51 0.54 25659

accuracy 0.56 50996

macro avg 0.57 0.57 0.56 50996

weighted avg 0.57 0.56 0.56 50996

In **Quadratic Aiscriminant Analysis** (QDA) we have got following results using the same settings. QDA have better ability to predict return cases (accuracy is 77%) comparing to LDA (accuracy is 51%).

Accuracies = [0.51600569 0.53046718 0.5249044 0.52666928 0.53166977 0.52804196 0.52715953 0.53804295 0.53039514 0.53431709]

Model Quadratic Discriminant Analysis

Mean Acc. Training 0.528767

Standard Deviation 0.00562223

Acc. Test 0.527159

Confusion Matrix Testing:

[[ 7113 18224]

[ 5889 19770]]

The model predicts the return case (case 1 ) which is our business case really well. It gets clearer from the below tabl

precision recall f1-score support

0 0.55 0.28 0.37 25337

1 0.52 0.77 0.62 25659

accuracy 0.53 50996

macro avg 0.53 0.53 0.50 50996

weighted avg 0.53 0.53 0.50 50996

* 1. **Neural Networks**

**Neural networks** use multiple hidden layers and non-linear activation functions to predict output of our state of return. Advantages of neural network are that they less formal training comparing do other models, they detect complex nonlinear dependence, and they can find interaction between predictors.

We were not able to perform the cross validation for Neural network model as it took a lot of time. But when we did the give us perfectly fitted result with accuracy of 57.9% with 16 hidden neurons being most appropriate.

Confusion Matrix Training:

[[48329 53780]

[32055 69818]]

Accurray Training: 0.5792030669372787

Confusion Matrix Testing:

[[11945 13435]

[ 8253 17363]]

Accurray Test: 0.5747117420974194

Hidden Neurons acctr accte

1.000 0.567 0.567

2.000 0.567 0.567

3.000 0.567 0.567

4.000 0.575 0.571

5.000 0.569 0.569

6.000 0.573 0.573

7.000 0.573 0.569

8.000 0.574 0.572

9.000 0.574 0.572

10.000 0.575 0.572

11.000 0.575 0.574

12.000 0.577 0.570

13.000 0.577 0.573

14.000 0.577 0.573

15.000 0.579 0.574

16.000 0.579 0.575

17.000 0.579 0.575

18.000 0.576 0.572

19.000 0.581 0.574

20.000 0.579 0.572

We get the robust model with high accuracy when the hyperparameter of hidden neuron layer is 16. Below is the accuracy

Hidden Neurons acctr accte

16.000 0.579 0.575

Confusion Matrix Training:

[[46041 56068]

[31044 70829]]

Accurray Training: 0.5729427106313302

Confusion Matrix Testing:

[[11342 14038]

[ 7942 17674]]

Accurray Test: 0.5689858028080633

* 1. **Support Vector Machines**

SVM) builds the map with margins between two outcomes as far apart as possible. We were not able to do cross validation with SVM but we tried to do it without cross validation. (The results of SVM model is perfectly fitted giving us with an average accuracy of 56.5 % )

Confusion Matrix Training:

[[7099 5055]

[5439 6688]]

Accurray Training: 0.5678102219842676

Confusion Matrix Testing:

[[1746 1276]

[1360 1689]]

Accurray Test: 0.565804645033767

* 1. **Ensemble model with Decision Tree and Naïve Bayes**

After going through all the models we decided to use the ensemble of Naïve Bayes and Decision tree to form our final model. These two model was selected because Naïve bayes very good results for our return prediction(case 1) while Decision tree also gave us good result for both 0 and 1 case, but fairly better result for return case 1 which is the business issue in hand. Also for complex models we received accuracy quite similar to these and hence it made sense to not over complicate the model and choose these two.

Following are the result for it when the weightage was 1 and 1 and hard voting. We get average results for our Business case Returns =1

Model Ensemble (equal, hard)

Mean Acc. Training 0.572168

Standard Deviation 0.00269147

Acc. Test 0.573927

Name: 2, dtype: object

Confusion Matrix Testing:

[[15351 10219]

[11509 13917]]

precision recall f1-score support

0 0.57 0.60 0.59 25570

1 0.58 0.55 0.56 25426

accuracy 0.57 50996

macro avg 0.57 0.57 0.57 50996

weighted avg 0.57 0.57 0.57 50996

Now when we again do hard voting with weight 1 for decision tree and 2 for naïve bayes( as it has the best accuracy for our return case 1). We get the better results for returns(case 1) which is the problem in hand

Comparison with Ensemble (weighting):

Model Ens. (weighted, hard)

Mean Acc. Training 0.518796

Standard Deviation 0.00245622

Acc. Test 0.519198

Name: 3, dtype: object

Confusion Matrix Testing:

[[ 5785 19785]

[ 4734 20692]]

precision recall f1-score support

0 0.55 0.23 0.32 25570

1 0.51 0.81 0.63 25426

accuracy 0.52 50996

macro avg 0.53 0.52 0.47 50996

weighted avg 0.53 0.52 0.47 50996

Now in the third scenario we kept the weightage same but did the soft voting(based on probability). The results were really good for the return (case=1) our business problem. Below is the result for this

Comparison with Ensemble (Soft Voting):

Model Ens. (weighted, soft)

Mean Acc. Training 0.538263

Standard Deviation 0.00405792

Acc. Test 0.537689

Name: 4, dtype: object

Confusion Matrix Testing:

[[ 7249 18321]

[ 5255 20171]]

precision recall f1-score support

0 0.58 0.28 0.38 25570

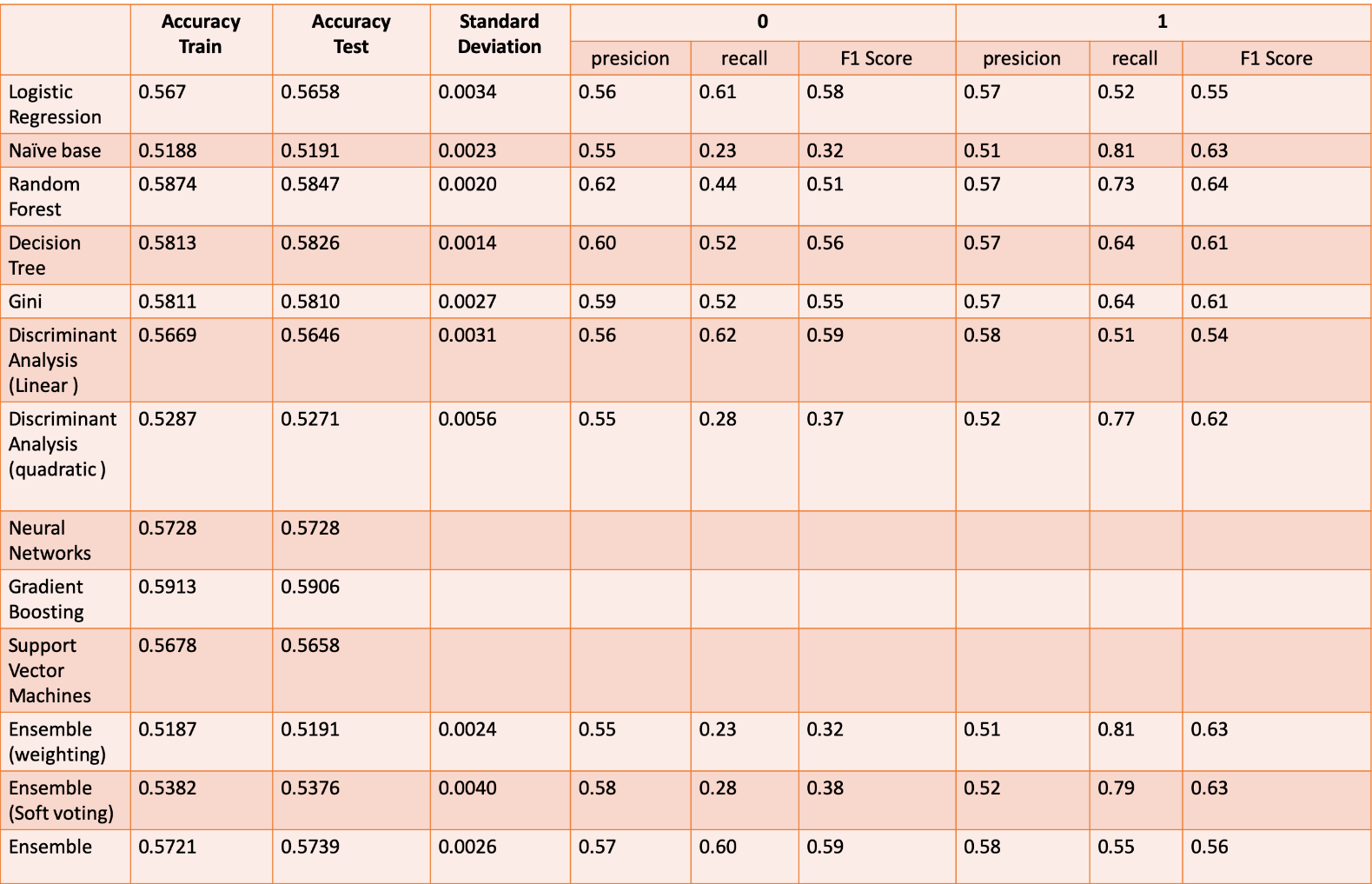
1 0.52 0.79 0.63 25426

accuracy 0.54 50996

macro avg 0.55 0.54 0.51 50996

weighted avg 0.55 0.54 0.51 50996

* 1. **Results of all the model**



* 1. **Which model to choose**

After going through results of all the models we would either choose either Ensemble with Naïve Bayes and Decision Tree or we can just go with Naïve Bayes or Quadratic Discriminant Analysis as in all these models we get the best results for returns( case 1) which is the business case in hand.

* 1. **Conclusion**

From the Gini model, we can see that price is the most significant feature that will influence whether the customer will return or not. Besides, Item ID, manufacturer ID, customer ID, age\_at\_order\_yrs, product\_age\_days are also important. Therefore, the online shop can use these feature on helping them build a model to predict whether the customer will return the order or not. To our case, if we combine with the significant features with our model, we can classified our customer on their behavior and predict the return probability, and help the online retailer to work on parameters like restriction on the basis of payment option, shipping costs, sizing guides.