

Project 36 : Santander product recommendation

avik423@outlook.com

Avik Ghosh

Problem Definition

Here problem is to recommend best product or products for users depending upon their or other similar user's past behaviours. To be more specific we have to find what users will purchase in the month of 28Th June 2016 in addition to what they already have till 28Th April 2015. To solve this problem a recommender system is required.

Business impacts of solving this problem

This recommender system recommends product/products to users depending upon their behaviours. It tries to get the best product or products for a user. While user is using previously purchased product he/she also checks other recommended products and as these products are recommended depending upon their or similar type of other customer's behaviour hence purchasing chance of a recommended product is significantly higher. This helps any organization to generate revenue without any intervention of other people. This system will try to find similar customers hence that will help to better organize different type of customer depending upon their behaviour and need. This also helps an organization to better serve their users and achieve customer satisfaction. By investing more time on organization page user will be more familiarized and that will increase future purchase rate from users.

Real-world impacts of solving this problem

Time is the most precious commodity in this century. User don't have time to check all the available products and their features to fit their needs. This type of system will help them to save their time by recommending proper products to them even before they search for it.

Dataset

1.Sources of the dataset:

The given dataset is provided by Banco Santander or Santander Bank.

Data is downloaded from this kaggle page: <https://www.kaggle.com/c/santander-product-recommendation/data>

Banco Santander: Banco Santander is the 16Th largest banking institution in the world and 4Th largest bank in Europe.

2.Data description

This is a data of 1.5 years of bank customer behaviours to predict what new products customers will purchase. The data is stored from 28Th January 2015 to 28Th May 2016.

There are 3 files inside provided data. They are

- train.csv - the training set
- test.csv - the test set
- sample_submission.csv - a sample submission file in the correct format

Description about data fields

Column name	Description
fecha_dato	The table is partitioned for this column
ncodpers	Customer unique code
ind_empleado	Employee index: A active, B ex employed, F filial, N not employee, P pasive
pais_residencia	Customer's Country residence
sexo	Customer's sex
age	Customer's Age
fecha_alta	The date in which the customer became as the first holder of a contract in the bank
ind_nuevo	New customer Index. 1 if the customer registered in the last 6 months.
antiguedad	Customer seniority (in months)
indrel	1 (First/Primary), 99 (Primary customer during the month but not at the end of the month)
ult_fec_cli_1t	Last date as primary customer (if he isn't at the end of the month)
indrel_1mes	Customer type at the beginning of the month ,1 (First/Primary customer), 2 (co-owner),P (Potential),3 (former primary), 4(former co-owner)
tiprel_1mes	Customer relation type at the beginning of the month, A (active), I (inactive), P (former customer),R (Potential)
indresi	Residence index (S (Yes) or N (No) if the customer country is the same than the bank country)
indext	Foreigner index (S (Yes) or N (No) if the customer's birth country is different than the bank country)
conyuemp	Spouse index. 1 if the customer is spouse of an employee
canal_entrada	channel used by the customer to join
indfall	Deceased index. N/S
tipodom	Addres type. 1, primary address
cod_prov	Province code (customer's address)
nomprov	Province name
ind_actividad_cliente	Customer Activity index (1, active customer; 0, inactive customer)
renta	Gross income of the household for customer

segmento	Customer segmentation: 01 - VIP, 02 - Individuals 03 - college graduated
ind_ahor_fin_ult1	Saving Account
ind_aval_fin_ult1	Guarantees
ind_cco_fin_ult1	Current Accounts
ind_cder_fin_ult1	Derivative Account
ind_cno_fin_ult1	Payroll Account
ind_ctju_fin_ult1	Junior Account
ind_ctma_fin_ult1	Plus particular Account
ind_ctop_fin_ult1	particular Account
ind_ctpp_fin_ult1	particular Plus Account
ind_deco_fin_ult1	Short-term deposits
ind_deme_fin_ult1	Medium-term deposits
ind_dela_fin_ult1	Long-term deposits
ind_ecue_fin_ult1	e-account
ind_fond_fin_ult1	Funds
ind_hip_fin_ult1	Mortgage
ind_plan_fin_ult1	Pensions
ind_pres_fin_ult1	Loans
ind_reca_fin_ult1	Taxes
ind_tjcr_fin_ult1	Credit Card
ind_valo_fin_ult1	Securities
ind_viv_fin_ult1	Home Account
ind_nomina_ult1	Payroll
ind_nom_pens_ult1	Pensions
ind_recibo_ult1	Direct Debit

**Above data description is taken from <https://www.kaggle.com/c/santander-product-recommendation/data>

Data Size:

File Name	Size	Number of rows	Number of columns	Description
train.csv.zip	214MB	13647309	48	This data contains all information to train the model. It contains all columns described above
test.csv.zip	12.3MB	929615	24	This data contains all information to test the model. It contains columns from <i>fetcha_dato</i> to

				<i>segmento</i>
sample_submission.csv.zip	2.28MB	929615	2	This contains a sample recommendation of products for all test users. It only contains 2 columns they are <i>ncodpers</i> & <i>added_products</i>

Many terms in this data is in Spanish language. To understand and process this data sometimes translation from Spanish to English is required.

Also the size of the data for the train and test is huge. If we are going to process this data using pandas it will not process with default set of parameters. Inside parameters of `read_csv()` one has to use `low_memory=False` then the data will be read by the system and it will take a lot of time to process it. To solve this problem one must use generators of python using `yield` and limit the data to a smaller chunk of datasets for data acquisition.

Following tools will be used in this problem:

Python: Overall programming language will be Python. It will be used for all coding purpose.

Pandas: This python library will be used for data analysis in this problem.

Plotly: This will be used to create beautiful plots for the data.

Matplotlib: This library will also be used to create plots for the problem.

Scikit-learn: This python library will be used for machine learning models.

Mean average precision

At present recommender systems are used everywhere in online. We are interacting with several of them everyday. While deciding a KPI for a recommender system one must think how a good recommender system works. A good recommender system always gives a customer some specific number of recommendations which is not overwhelmingly long or not too short. Also a recommender system must put the recommendations in a sequence from best to worst order to generate maximum user interaction and sale. While selecting a KPI for this problem, the same points are also taken into consideration. As per the selected KPI from Kaggle competition of this problem was Mean Average Precision or MAP. It is also used here as the KPI.

Mean average precision is the average of average precisions of number of users' recommendations. Here $\text{precision}(P)$ is represented as below:

$$P = \frac{TP}{TP+FP} \quad (TP=\text{True positive}, FP=\text{False positive})$$

For the recommender system P can be represented as:

$$P = \frac{\text{Number of relevant recommendations}}{\text{Total number of items recommended}}$$

Precision metric is used mainly for information retrieval which is most relevant. For this problem we need the same for several users and for several products provided by Bank.

Hence Average Precision(AP) is used to calculate that.

Average Precision can be represented as:

$$AP@N = \frac{1}{m} \sum_{k=1}^n P(k) r(k)$$

Where,

m=Number of relevant recommendations
and m is not zero

N= Number of recommendations system is making

k = each iteration inside the precision list

P(k) = Precision in kth iteration

r(k) = Relevance in kth iteration if it is relevant then r(k)=1 else it is 0.

Let's take an example with a similar list but element position of this binary list is different from each other

product_recommendation_list1=[1,1,0,0]

product_recommendation_list2=[0,1,1,0]

product_recommendation_list2=[0,0,1,1]

$$\text{Average precision for } [1,1,0,0] = \frac{\frac{1}{1} + \frac{2}{2} + \frac{2}{3} + \frac{2}{4}}{4} = 0.79 \text{ (approx)}$$

$$\text{Average precision for } [0,1,1,0] = \frac{\frac{0}{1} + \frac{1}{2} + \frac{2}{3} + \frac{2}{4}}{4} = 0.42 \text{ (approx)}$$

$$\text{Average precision for } [0,0,1,1] = \frac{\frac{0}{1} + \frac{0}{2} + \frac{1}{3} + \frac{2}{4}}{4} = 0.21 \text{ (approx)}$$

It is obvious from the above example that if relevant products are recommended first then Average Precision rewards with more score. For this problem it is also required to give relevant recommendations first then recommend other products.

Now the Mean term comes as the above process is done for a number of users and Mean of their average precision is required.

Hence Mean Average Precision is represented as:

$$\text{MAP@N} = \frac{1}{U} \times \sum_{u=1}^U (\text{AP@N})$$

$$\text{MAP@N} = \frac{1}{U} \times \sum_{u=1}^U \frac{1}{m} \sum_{k=1}^n P(k) r(k)$$

Where,

m=Number of relevant recommendations and m is not zero

N= Number of recommendations system is making

k = each iteration inside the precision list

P(k) = Precision in kth iteration

r(k) = Relevance in kth iteration if it is relevant than r(k)=1 else it is 0.

U = Number of users and U is not zero.

u = each user from 1 to U.

Why other metrics are not used here?

Accuracy and error based metrics are not used here as they all are used to check already decided items and their correctness. But here in this recommender problem ranking as well as relevant recommendation both are required hence not using MAE(Mean absolute error),MSE(Mean squared error) or RMSE(Root mean squared error).

There are decision support metrics like precision, recall that are used here but also other factors like average of precision is used here as well as ranking is also taken into consideration . More over decision metrics takes the entire dataset they are does target or not targeted to Top N recommendations.

The metric Mean Average Precision is a rank aware evaluation based metric where decision making as well as arranging of items both are given preference. But there are some other rank aware evaluation based metric available like Mean Reciprocal Rank,Normalized discounted cumulative gain.

Mean reciprocal rank

Mean reciprocal rank puts more emphasis on the rank of the 1st relevant recommended item.

Let's take an example:

rec1=[1,0,0,1], rec2=[0,1,0,1], rec3=[0,0,1,1]

Here, the reciprocal rank for rec1=1/1

the reciprocal rank for rec2=1/2

the reciprocal rank for rec3=1/3

Hence mean reciprocal rank=(1+1/2+1/3)/3=0.611

This metric is best when one wants to recommend a user only a single product because this metric does not give any importance to the other element's relevance after the first one. Here, the problem needs more than one product in recommendation; hence this metric is not used.

Normalized discounted cumulative gain(NDCG)

Both MAP and NDGC put relevant recommendation first and also give specific number of recommendation required by the system but NDCG gives a more importance in order of relevant recommendations. Like most relevant data should be given first then far less relevant data should be given at last irrelevant data should be given. But there is a problem with NDCG that it suffers a lot if there is incomplete recommendation sequences which is the case for almost every real life recommender system. If incomplete data ratings are replaced with 0 then NDGC will take it as irrelevant.

MAP can handle binary recommendation data very well where 0 means irrelevant and 1 means relevant. It gives more value if relevant recommendations are provided before. Conversely it gives more weight to the error that happens first in comparison to less weight to the errors that happen later on.

Mean Average Precision does not work well if ratings are non-binary. If ratings are 1 to 5 or 1 to 10 then MAP cannot be applied. Even though it is applied using a condition like rating ≥ 3 is 1 and rating < 3 is 0, even then data received from this metric will not be useful as rating 3 and rating 5 are not the same similarly rating 0 and rating 2 are not the same.

Mean Average Precision is used in object detection. In object detection problem there could be different objects of different classes. In this situation both classification and detecting position of object needs to be evaluated. As standard precision metric can not be used here hence Mean Average Precision is used.

This problem is mapped to a multi level classification problem. In the given dataset there are $25-48=23$ products with binary ratings and each user's recommendations will be represented by multiple label throughout each row of user.

Multi level classification problem can be solved by various ways. Most easy way to solve this problem will be taking each product label as a separate binary classification problem.

Another way to solve this problem is to use classifier chains. First we take users as input and take first product label as the target variable in the next step along with user and first product label both will be taken as input and second product label as target variable. In this way in each each step we shall solve a different binary classification problem.

There are other ways like adaptive algorithms(multi label KNN) also ensemble models can be used.

Dataset label and output variable analysis

This problem refers to a multi label recommendation problem. Given data includes 24 labels for recommendation. They are following:

ind_ahor_fin_ult1 (p1)	Saving Account
ind_aval_fin_ult1 (p2)	Guarantees
ind_cco_fin_ult1 (p3)	Current Accounts
ind_cder_fin_ult1 (p4)	Derivative Account
ind_cno_fin_ult1 (p5)	Payroll Account
ind_ctju_fin_ult1 (p6)	Junior Account
ind_ctma_fin_ult1 (p7)	Plus particular Account
ind_ctop_fin_ult1 (p8)	particular Account
ind_ctpp_fin_ult1 (p9)	particular Plus Account
ind_deco_fin_ult1 (p10)	Short-term deposits
ind_deme_fin_ult1 (p11)	Medium-term deposits
ind_dela_fin_ult1 (p12)	Long-term deposits
ind_ecue_fin_ult1 (p13)	e-account
ind_fond_fin_ult1 (p14)	Funds
ind_hip_fin_ult1 (p15)	Mortgage
ind_plan_fin_ult1 (p16)	Pensions
ind_pres_fin_ult1 (p17)	Loans
ind_reca_fin_ult1 (p18)	Taxes
ind_tjcr_fin_ult1 (p19)	Credit Card
ind_valo_fin_ult1 (p20)	Securities
ind_viv_fin_ult1 (p21)	Home Account
ind_nomina_ult1 (p22)	Payroll
ind_nom_pens_ult1 (p23)	Pensions
ind_recibo_ult1 (p24)	Direct Debit

For better interpretation renamed and used them as p1 to p24.

Except p22 and p23 all labels has only 2 types of values. They are 0 and 1 and they all are integer. p22 and p23 is having float values though they are also having only 0.0 and 1.0.

As per features renta,age,antiguedad,ncodpers,cod_prov are having numeric values rest of them are all categorical data except fecha_dato and fecha_alta. These two columns are datetime category. There is a column named nomprov which is a object type data.

Here columns are not in their proper datatype.

Data type of all the features columns:

fecha_dato	object
ncodpers	int64
ind_empleado	object
pais_residencia	object
sexo	object
age	object
fecha_alta	object
ind_nuevo	float64
antiguedad	object
indrel	float64
ult_fec_cli_1t	object
indrel_1mes	object
tiprel_1mes	object
indresi	object
indext	object
conyuemp	object
canal_entrada	object
indfall	object
tipodom	float64
cod_prov	float64
nomprov	object

ind_actividad_cliente	float64
renta	float64
segmento	object

Univariate feature analysis:

In univariate feature analysis only one dependable variable or feature can be used. It generally helps to analyse the pattern of the data also it helps to define and summarize the data.

Univariate feature analysis can be done using following options:

Frequency distribution table:

It takes all data and shows the number of time each data occurs. It helps to give a brief idea about the data. It also helps in data cleaning and understanding the common statistics.

Bar charts:

Bar chart is presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent. It is very useful when comparing with frequency along with class labels. Specially when analysing categorical features.

Histograms:

Histograms are just like bar charts but they have a difference. Here in place of class categories bins are used. Hence this type of chart is more efficient while used for continuous data.

Pie charts:

Pie charts represent the class labels by using slices proportional to the frequency of the selected column variables. The whole pie represents 100%.

Here in this problem, frequency table is used mainly. As this problem has 24 class labels and each of them has again 2 type of data. Hence used frequency table to check data pattern. Using other methods will be difficult to manage the diagrams as each of the feature will need 24 plots each for a product.

Bivariate feature analysis:

In bivariate feature analysis two features are taken into consideration it is generally used to check relationship between two features.

Bivariate feature analysis can be done using following options:

Scatter plots:

Scatter plots are created by taking two features one as x co-ordinate and another as y co-ordinate. It helps to detect if the two of them are linear or non linear also it helps to check the relationship between these two features.

Correlation plots:

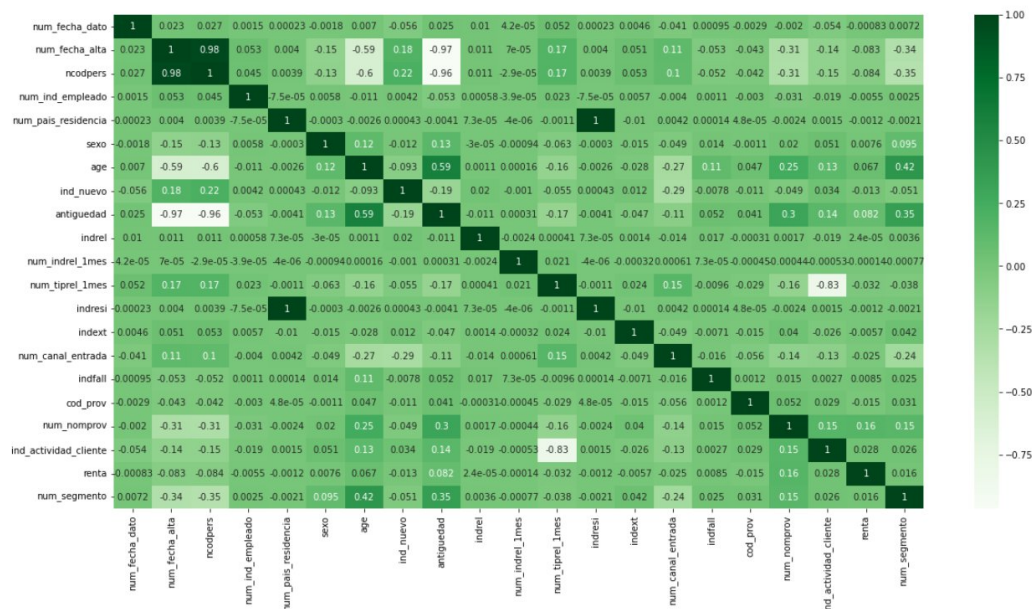
Here in this problem correlation plots are used. As it is easier to interpret a huge set of features as the given problem dataset is having more than 20 features. Scatter plot is not used here as to cover all the variables a lot of scatter plots are required as and it is hard to manage even after drawing those huge number of scatter plots.

Here correlation refers to:

$$\text{corr}(X, Y) = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y}$$

Where ,

$\text{cov}(X, Y)$ refers to covariance of X and Y and σ_X, σ_Y refers to standard deviation of X and Y.



Multivariate feature analysis:

Multivariate analysis is used when there are more than 2 features are taken together for feature analysis. It is hard to analysis this type data for a human. Hence Some specific techniques are used to do this type of analysis.

Cluster analysis:

It is a task of grouping where similar type of objects are included in a group called cluster. A dataset with multiple features can be differentiated into various groups by using this concept.

Principal component analysis(PCA):

PCA is used to reduce the dimensionality of data. In this process original features are converted into a new set of features which are known as principal components. After that we can use scatter plot to check the dataset features and their relationships.

In this problem PCA is used for multivariate analysis as there are 17 features after data cleaning and selecting important features. Hence first used PCA to decrease the dimension of data to 2 then tried scatter plot with all products.

Encoding features:

In this problem we already have 24 rows and after checking correlation and removing unnecessary columns we have 17 columns. They are:

fecha_dato,ncodpers,ind_empleado,pais_residencia,sexo,age,fecha_alta

,ind_nuevo,antiguedad,indrel,indrel_1mes,tiprel_1mes,indresi,indext,

canal_entrada,indfall,cod_prov,nomprov,ind_actividad_cliente,renta,segmento

Here fecha_dato and fecha_alta are datetime data which is not a numerical data hence encoded it by stacking yearmonthday format for each value by creating a specific function named converting_dates. It converts a date like 2018-09-28 to 20180928.

Here ncodpers,age,renta,cod_prov are all already in numeric format.

Categorical data with 2 categories like sexo are encoded with label encoder. One hot encoding is not used to avoid extra dimensions. Categorical data with more than 2 category(ind_empleado) is encoded with a modified frequency encoder where we are diving frequency by number of total points as a representation of that specific category.

In this problem, 24 labels are converted into one single class column and by converting that into a single class column the multilabel classification problem is converted into a multiclass classification problem.

Base Line models:

Base Line models are the simple models that provides reasonable results on a problem at a small amount of time and without vast expertise. Though baseline models are not perfect. It does not gives the best result instead it helps to reach there step by step. Logistic Regression is a base line model for this problem.

Metrics:

Metrics used here are Accuracy, Precision, Recall and F1 score. All of these scores are vastly dependent upon confusion matrix. Confusion matrix is created on True Positive(TP), True Negative(TN), False Positive(FP), False Negative(FN).

Accuracy:

Accuracy gives the overall accuracy of the model i.e fractions of the total points are correctly classified by the classifier among all points. To calculate accuracy following formula is used:

$$Accuracy = \frac{TP+TN}{(TP+TN+FP+FN)}$$

1-Accuracy:

This gives how much error is model having overall while predicting classes. It is just opposite of Accuracy. This can be used as a loss function.

Precision:

Precision tells us among all the predicted positive points how many of them are actually positive. Following formula is used to calculate the precision:

$$Precision = \frac{TP}{(TP+FP)}$$

Recall:

Recall tells us among all the true positive points how many of them are actually positively predicted. Following formula is used to calculate the recall:

$$Recall = \frac{TP}{(TP + FN)}$$

F1- score:

F1 score manages both metric precision and recall i.e if F1 score is high that means the model is having both high precision and recall . Following formula is used to calculate the F1 Score:

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

After pre-processing the data and creating environment without using any tuning and optimizations comparing several models and got the following result:

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.7270	0.8677	0.3385	0.6706	0.6737	0.3703	0.4012	3.1550
gbc	Gradient Boosting Classifier	0.7259	0.8679	0.3295	0.6627	0.6672	0.3600	0.3945	121.6960
rf	Random Forest Classifier	0.7126	0.8478	0.3057	0.6363	0.6536	0.3264	0.3588	6.4980
lda	Linear Discriminant Analysis	0.6904	0.8295	0.3019	0.6138	0.6369	0.2980	0.3186	2.4260
et	Extra Trees Classifier	0.6876	0.8233	0.2905	0.6132	0.6374	0.2928	0.3122	11.5840
ridge	Ridge Classifier	0.6846	0.0000	0.1721	0.5518	0.5825	0.1336	0.2070	0.1700
dummy	Dummy Classifier	0.6686	0.5000	0.1000	0.4471	0.5358	0.0000	0.0000	0.0820
nb	Naive Bayes	0.6657	0.7383	0.2074	0.5594	0.5960	0.2292	0.2499	0.2850
ada	Ada Boost Classifier	0.6629	0.6966	0.2589	0.6074	0.6146	0.2473	0.2715	3.0090
lr	Logistic Regression	0.6613	0.7050	0.1180	0.4899	0.5572	0.0797	0.1164	8.3040
knn	K Neighbors Classifier	0.6298	0.6783	0.1533	0.5329	0.5695	0.1379	0.1506	2.5410
dt	Decision Tree Classifier	0.6216	0.6634	0.3349	0.6254	0.6234	0.2996	0.2996	0.7050
svm	SVM - Linear Kernel	0.4009	0.0000	0.1029	0.3741	0.3377	0.0077	0.0176	8.2900
qda	Quadratic Discriminant Analysis	0.0713	0.5111	0.1237	0.4903	0.0598	0.0134	0.0200	1.5850

CPU times: total: 1min 7s
Wall time: 28min 35s

We are getting this result while using raw data. Here lightgbm is giving best result with decent accuracy, precision and f1 score. Though this is not the best way to represent the model and its accuracy. Also the model is not optimized also data is not scaled and we have not applied any transformations.

We can see here that base line models like logistic regression, Knn etc are getting accuracy like 53% which is not great but more than 50%. Whereas the best model with this observation is lightgbm which is a gradient boosting framework though it is faster than gradient boosting classifier also gives better accuracy and trains very quickly.

Now to optimize this solution we shall use normalization. Normalization is a scaling process for data set. It makes sure that all different variable are measured in same scale. It is used mainly when data sets are varying in scale. It makes a lot of difference in model performance.

Here we are using zscore as the normalization method. Z score is represented by following:

$Z = \frac{x - \mu}{\sigma}$, Here x is the original feature vector, μ is mean and σ means standard deviation. Z score is vastly used in machine learning problems.

Let's take a look at the raw data without scaling applied:



As we can see here antigüedad starts from 0 and ends at 250 whereas age starts at 0 but ends at 120. There is a huge difference between upper limit of two feature scale. Now after applying ZScore we get:



Here we can see the data of age starts from -2 and ends at 4 whereas data of antigüedad starts from -1.5 and ends at 3.0 distance between two upper limit scale is minimized a lot.

We shall also use Transformation of features to improve our machine learning solution. As normalization scales the data with new limits to reduce the impact of magnitude in the variance , transformation transforms the data to make it a normal or approximately normal distribution. There are several process to transform the data. Yeo-johnson , Quantile transformation etc. Quantile transformation is non-linear and may distort linear co-relations between variables measured at the same scale. Hence using Yeo-johnson method of transformation. Also we are using 100000 data to minimize time for training. After applying Yeo-johnson transformation on clean data it looks like this:



It seems much more separable than raw data scatter plot. Hence in our next instance using both scaling and transformation of data.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
gbc	Gradient Boosting Classifier	0.7269	0.8680	0.3303	0.6585	0.6682	0.3589	0.3943	60.5570
lightgbm	Light Gradient Boosting Machine	0.7260	0.8672	0.3340	0.6653	0.6716	0.3633	0.3951	1.3490
rf	Random Forest Classifier	0.7123	0.8471	0.3058	0.6350	0.6532	0.3241	0.3560	5.0350
lr	Logistic Regression	0.7104	0.8437	0.2699	0.6108	0.6287	0.2682	0.3263	26.5770
svm	SVM - Linear Kernel	0.7044	0.0000	0.2449	0.5629	0.5975	0.2050	0.2972	1.4100
ridge	Ridge Classifier	0.7005	0.0000	0.2230	0.5664	0.6007	0.1912	0.2756	0.1260
lda	Linear Discriminant Analysis	0.6929	0.8317	0.3117	0.6165	0.6410	0.3083	0.3273	1.6100
et	Extra Trees Classifier	0.6915	0.8253	0.2923	0.6140	0.6387	0.2943	0.3157	7.6880
knn	K Neighbors Classifier	0.6788	0.7562	0.2671	0.5982	0.6234	0.2592	0.2803	18.8550
dummy	Dummy Classifier	0.6698	0.5000	0.1000	0.4486	0.5373	0.0000	0.0000	0.0650
ada	Ada Boost Classifier	0.6519	0.6885	0.2731	0.6113	0.6087	0.2486	0.2766	1.6410
dt	Decision Tree Classifier	0.6238	0.6633	0.3390	0.6268	0.6252	0.3017	0.3017	0.6320
qda	Quadratic Discriminant Analysis	0.4032	0.5583	0.1340	0.5204	0.4381	0.0700	0.0746	1.0130
nb	Naive Bayes	0.0533	0.6867	0.1858	0.5960	0.0230	0.0245	0.0401	0.2630

CPU times: total: 5min 13s
Wall time: 26min 55s

We can see the performance jump of at least 5% from accuracy column of models like logistic regression,svm,ridge. Though Gradient boosting,light gradient boosting and random forest has not changed much.

In next test polynomial features are added along with transformation and scaling and performance for top 5 models are improved a little. It is not a significant change in performance.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
gbc	Gradient Boosting Classifier	0.7269	0.8680	0.3302	0.6577	0.6681	0.3589	0.3943	65.2770
lightgbm	Light Gradient Boosting Machine	0.7260	0.8672	0.3340	0.6653	0.6716	0.3633	0.3951	1.5420
rf	Random Forest Classifier	0.7126	0.8474	0.3066	0.6357	0.6534	0.3241	0.3564	5.0130
lr	Logistic Regression	0.7105	0.8437	0.2699	0.6112	0.6287	0.2683	0.3264	27.1900
svm	SVM - Linear Kernel	0.7054	0.0000	0.2436	0.5557	0.5963	0.2011	0.2986	1.4530
ridge	Ridge Classifier	0.7005	0.0000	0.2230	0.5664	0.6007	0.1912	0.2756	0.1350
lda	Linear Discriminant Analysis	0.6929	0.8317	0.3117	0.6165	0.6410	0.3083	0.3273	1.6410
et	Extra Trees Classifier	0.6909	0.8258	0.2913	0.6134	0.6383	0.2935	0.3146	7.7070
knn	K Neighbors Classifier	0.6788	0.7562	0.2671	0.5982	0.6234	0.2592	0.2803	19.0250
dummy	Dummy Classifier	0.6698	0.5000	0.1000	0.4486	0.5373	0.0000	0.0000	0.0590
ada	Ada Boost Classifier	0.6519	0.6885	0.2731	0.6113	0.6087	0.2486	0.2766	1.7890
dt	Decision Tree Classifier	0.6251	0.6642	0.3396	0.6275	0.6262	0.3031	0.3032	0.6590
qda	Quadratic Discriminant Analysis	0.4665	0.5578	0.1289	0.5226	0.4732	0.0706	0.0774	1.0140
nb	Naive Bayes	0.0533	0.6867	0.1858	0.5960	0.0230	0.0245	0.0401	0.2630

CPU times: total: 5min 28s
Wall time: 27min 31s

In the next test instance using transformation, normalization and linear pca with number of features to 10. PCA or principal component analysis is a dimensionality reduction technique where pca can create upto 2 features to represent all the features of the dataset. In this way various classes can be drawn in a 2d plot and similar points can be separated in a cluster. Actually what PCA does is , it finds the best fitting line by maximizing the sum of the squared distances from the projected points to the origin. And best fitting line that goes through origin is created which is called PC1, PC2 is calculated just by making a perpendicular on PC1 that also goes through origin and so on. Once all PC's figured out eigenvalues can be used which are sum of squared distances to determine the proportion of variation that each PC accounts for.

Got the following results after using linear pca:

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
gbc	Gradient Boosting Classifier	0.7046	0.8477	0.2347	0.6046	0.6161	0.2455	0.3065	81.7560
lightgbm	Light Gradient Boosting Machine	0.7040	0.8467	0.2647	0.6232	0.6318	0.2773	0.3213	3.0850
rf	Random Forest Classifier	0.7024	0.8405	0.2609	0.6192	0.6327	0.2803	0.3195	7.2870
lr	Logistic Regression	0.6956	0.8242	0.2186	0.5544	0.6022	0.2081	0.2710	4.0690
et	Extra Trees Classifier	0.6951	0.8331	0.2634	0.6093	0.6306	0.2767	0.3083	3.6050
lda	Linear Discriminant Analysis	0.6890	0.8239	0.2208	0.5582	0.6035	0.2180	0.2641	0.1060
svm	SVM - Linear Kernel	0.6876	0.0000	0.2098	0.5376	0.5920	0.1927	0.2504	0.2140
ridge	Ridge Classifier	0.6853	0.0000	0.1562	0.5243	0.5767	0.1142	0.2015	0.0330
nb	Naive Bayes	0.6778	0.8219	0.2476	0.5808	0.6132	0.2495	0.2732	0.0370
knn	K Neighbors Classifier	0.6691	0.7603	0.2558	0.5890	0.6176	0.2559	0.2718	1.0810
dummy	Dummy Classifier	0.6681	0.5000	0.1000	0.4464	0.5352	0.0000	0.0000	0.0320
ada	Ada Boost Classifier	0.6345	0.6952	0.2219	0.5500	0.5718	0.1727	0.2000	1.7980
qda	Quadratic Discriminant Analysis	0.5980	0.8204	0.3099	0.6368	0.6035	0.2823	0.2917	0.0540
dt	Decision Tree Classifier	0.5969	0.6460	0.2992	0.6049	0.6008	0.2609	0.2610	0.4040

CPU times: total: 6min 59s

Wall time: 24min 13s

Used another test instance with 2 PC's but results were even worse hence not using PCA in this problem as without using PCA performance is better.

In overall all test instances three models always performed better. They are Gradient Boosting classifier, Light Gradient Boosting Machine and Random Forest. Gradient Boosting classifier gives best result but takes more time than any 3 of the models. Light Gradient Boosting Machine takes very less train time in comparison to Gradient Boosting classifier and gives almost similar performance.

For next phase will use the setup with normalization and transformations.

After doing transformation and normalization using zscore and training with full data the accuracy got increased and gradient boosting classifier, random forest classifier and logistic regression is performing well they all are above 74% accuracy report is given below:

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
gbc	Gradient Boosting Classifier	0.7694	0.9165	0.3772	0.7235	0.7281	0.4253	0.4433	384.5770
rf	Random Forest Classifier	0.7577	0.9042	0.3565	0.7069	0.7218	0.4052	0.4187	26.5790
lr	Logistic Regression	0.7483	0.8949	0.3016	0.6761	0.6879	0.3258	0.3574	111.5510
lightgbm	Light Gradient Boosting Machine	0.7466	0.8899	0.3578	0.7037	0.7142	0.3951	0.4073	6.6060
svm	SVM - Linear Kernel	0.7411	0.0000	0.2754	0.6388	0.6690	0.2788	0.3213	4.3670
ridge	Ridge Classifier	0.7401	0.0000	0.2539	0.6381	0.6548	0.2161	0.2860	0.5170
et	Extra Trees Classifier	0.7398	0.8901	0.3423	0.6920	0.7100	0.3775	0.3858	217.1630
lda	Linear Discriminant Analysis	0.7201	0.8794	0.3868	0.7110	0.7123	0.4119	0.4126	8.8590
knn	K Neighbors Classifier	0.7182	0.8284	0.3084	0.6706	0.6889	0.3263	0.3335	129.6510
dummy	Dummy Classifier	0.7142	0.5000	0.1000	0.5100	0.5951	0.0000	0.0000	0.1700
qda	Quadratic Discriminant Analysis	0.7141	0.5002	0.1001	0.5539	0.5952	0.0005	0.0083	6.7900
dt	Decision Tree Classifier	0.6948	0.7087	0.3754	0.6966	0.6957	0.3614	0.3614	2.9860
ada	Ada Boost Classifier	0.6823	0.6974	0.1773	0.6009	0.6333	0.2375	0.2559	9.7340
nb	Naive Bayes	0.0335	0.7876	0.1947	0.7355	0.0157	0.0141	0.0235	1.2000

Even precision and F1 score are decent. Now using errors to increase the feature and trying to create a relation between features to increase the performance of the model.

For that we are going to first count each label frequency just like this:

```
#Loading other set of data using pickle
import pandas as pd
df1=pd.read_pickle("prep_data.pkl")
```

```
final_train.labels.value_counts()
```

```
00100000000000000000000000000000    280309
00100001000000000000000000000000    33000
00100000000000000000000000000001    20076
00000001000000000000000000000000    19651
00100000000010000000000000000000    10835
00000000000010000000000000000000    6836
00000100000000000000000000000000    6156
00001000000000000000000000000000    5440
000100000000000000000000000000111    5369
00100000100000000000000000000000    4818
Name: labels, dtype: int64
```

We shall detect the class with most frequency and encode it as 1 and all other class as 0 just like below table:

_dato	num_fecha_alta	ncodpers	num_ind_employed	num_pais_residencia	sexo	age	ind_nuevo	antiguedad	...	index	num_canal_entrada	indfall	tipodom	cod_prov	num_nomprov	ind_actividad_cliente	renta	num_segmento	label
50528	20060722	628950	0.999334	0.999993	1	46	0	118	...	0	0.237417	0	1.0	11.0	0.020825	0	81130.08	0.606865	0
50528	20141110	1351459	0.999334	0.999993	0	21	0	18	...	0	0.293058	0	1.0	33.0	0.018374	0	95542.17	0.351804	0
50528	20140915	1307576	0.999334	0.999993	0	22	0	20	...	0	0.293058	0	1.0	27.0	0.005181	1	61204.71	0.351804	0
50528	20131128	1226711	0.999334	0.999993	0	23	0	30	...	0	0.293058	0	1.0	28.0	0.359795	0	206361.81	0.351804	0
50528	20141003	1323250	0.999334	0.999993	0	23	0	19	...	0	0.293058	0	1.0	47.0	0.019660	0	128073.93	0.351804	0
50528	20140919	1311909	0.999334	0.999993	0	22	0	20	...	0	0.293058	0	1.0	6.0	0.013722	0	67890.27	0.351804	0
50528	20020124	321623	0.999334	0.999993	1	64	0	172	...	0	0.237417	0	1.0	28.0	0.359795	0	91748.67	0.606865	1

Now we shall feed this into a model and predict the labels also calculate proba scores and using proba score to distinguish correctly classified points with high and low confidence and wrongly classified points with high and low confidence and then again feed that to another machine learning model. We have used here random forest. And again checking proba score and as we are using 4 class like correct high and low same as wrong high and low hence we shall get 4 proba score now we shall use them as four numeric features in our model . Which also increases performance of top models by 1-2%. The result is given below:

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.7601	0.9060	0.3549	0.7088	0.7217	0.4044	0.4202	26.7210
lightgbm	Light Gradient Boosting Machine	0.7549	0.8945	0.3607	0.7058	0.7195	0.4037	0.4163	6.6720
lr	Logistic Regression	0.7487	0.8964	0.3033	0.6768	0.6898	0.3295	0.3601	124.4240
et	Extra Trees Classifier	0.7450	0.8942	0.3376	0.6928	0.7107	0.3771	0.3887	259.6280
ridge	Ridge Classifier	0.7425	0.0000	0.2585	0.6429	0.6599	0.2344	0.2997	0.5440
svm	SVM - Linear Kernel	0.7423	0.0000	0.2704	0.6351	0.6674	0.2658	0.3152	4.6890
lda	Linear Discriminant Analysis	0.7186	0.8809	0.3981	0.7203	0.7159	0.4224	0.4234	9.3480
knn	K Neighbors Classifier	0.7181	0.8282	0.3091	0.6711	0.6889	0.3269	0.3340	157.0470
dummy	Dummy Classifier	0.7142	0.5000	0.1000	0.5100	0.5951	0.0000	0.0000	0.1520
qda	Quadratic Discriminant Analysis	0.7106	0.5059	0.1020	0.5680	0.5975	0.0130	0.0279	6.0320
ada	Ada Boost Classifier	0.6982	0.6973	0.1780	0.6014	0.6433	0.2463	0.2626	10.5220
dt	Decision Tree Classifier	0.6947	0.7092	0.3764	0.6972	0.6959	0.3622	0.3622	3.1820
gbc	Gradient Boosting Classifier	0.6252	0.8345	0.3216	0.6515	0.5923	0.3426	0.3574	403.5240
nb	Naive Bayes	0.0335	0.7893	0.1940	0.7345	0.0155	0.0140	0.0235	1.2460

CPU times: total: 29min 3s

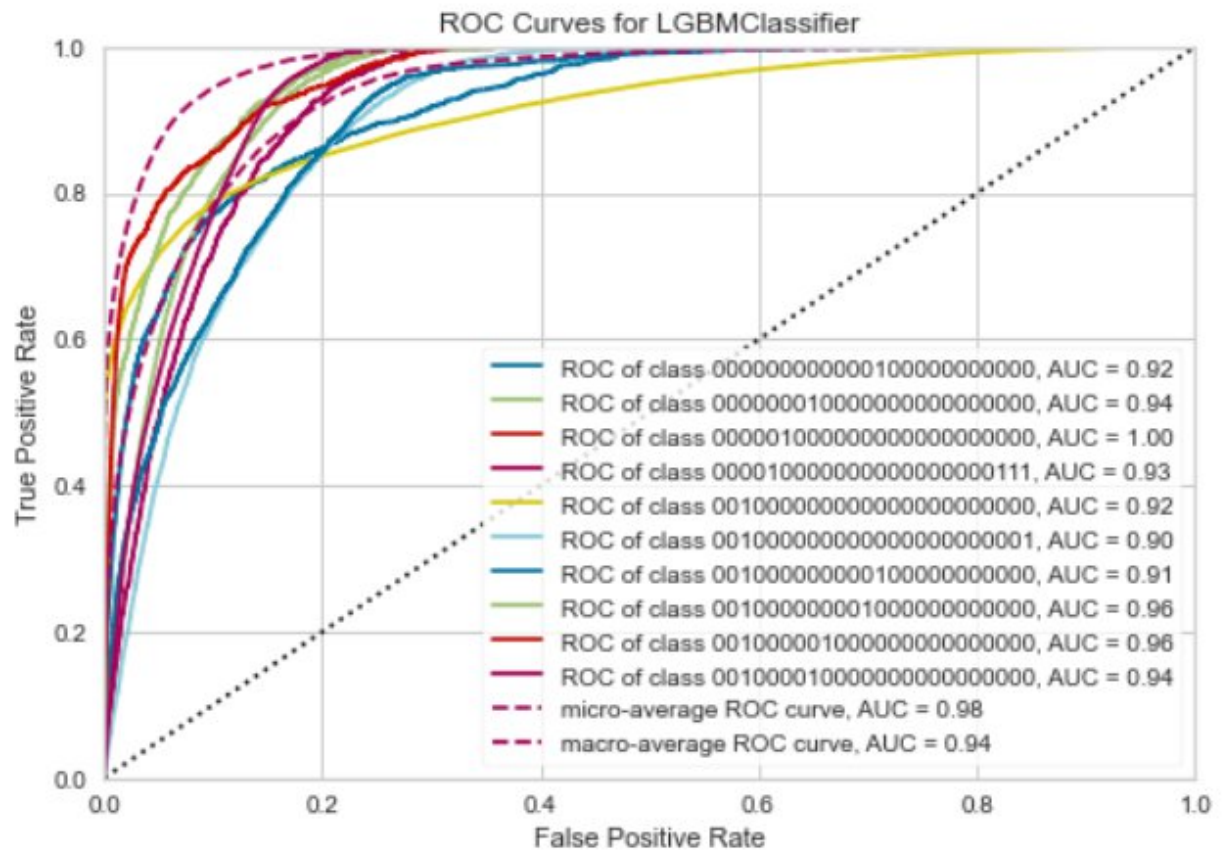
Even logistic regression is performing pretty well. As random forest and lightgbm are performing very well hence we shall select this 2 models and do further testing.

After tuning hyper parameters of lightgbm the performance got increased by even 2%

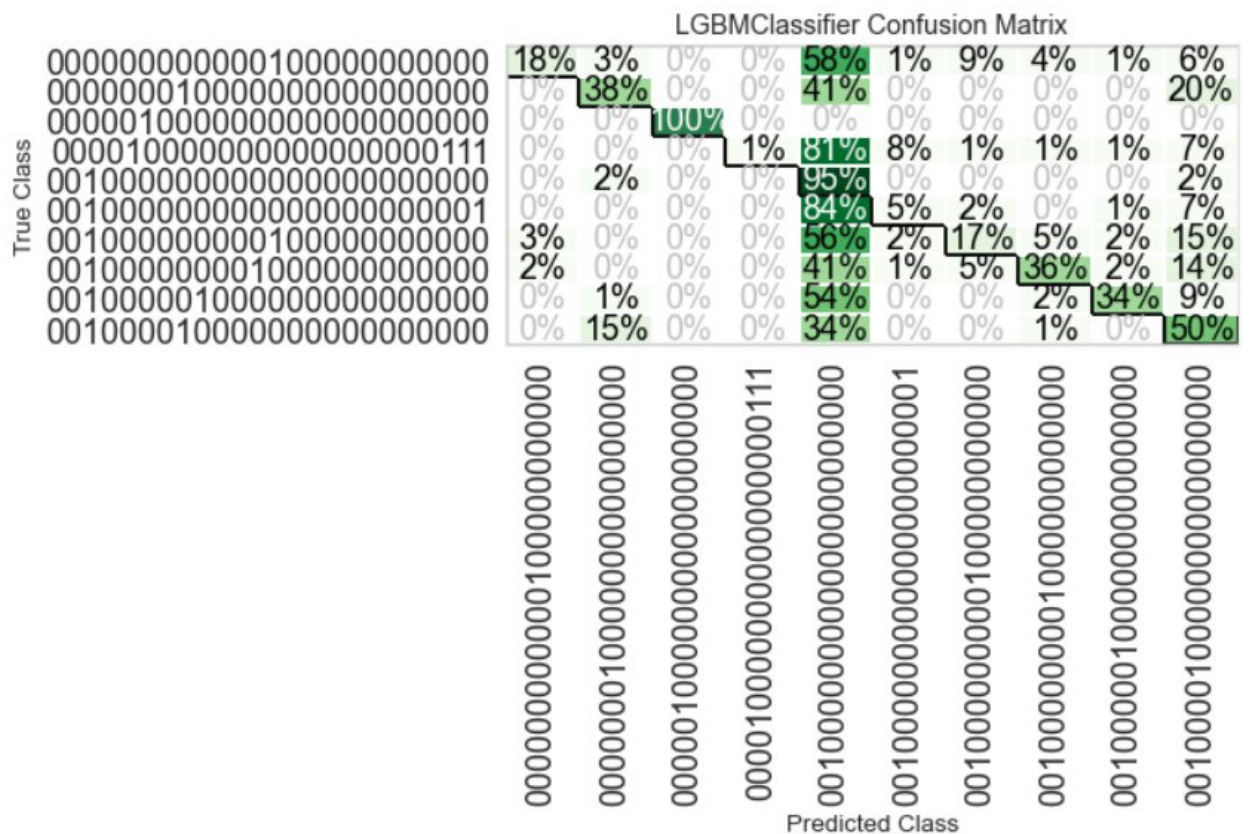
	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7702	0.9194	0.3892	0.7222	0.7333	0.4340	0.4496
1	0.7714	0.9187	0.3904	0.7295	0.7349	0.4353	0.4516
2	0.7697	0.9184	0.3944	0.7265	0.7337	0.4337	0.4490
3	0.7714	0.9182	0.3919	0.7272	0.7347	0.4363	0.4522
4	0.7709	0.9193	0.3930	0.7260	0.7340	0.4337	0.4501
5	0.7677	0.9180	0.3915	0.7192	0.7304	0.4268	0.4425
6	0.7691	0.9191	0.3829	0.7212	0.7313	0.4292	0.4454
7	0.7698	0.9185	0.3886	0.7235	0.7324	0.4307	0.4471
8	0.7717	0.9189	0.3896	0.7281	0.7355	0.4380	0.4536
9	0.7693	0.9167	0.3838	0.7250	0.7320	0.4275	0.4444
Mean	0.7701	0.9185	0.3895	0.7248	0.7332	0.4325	0.4485
SD	0.0012	0.0008	0.0035	0.0031	0.0016	0.0036	0.0034

CPU times: total: 2min 7s

Even precision, recall and f1 scored increased a lot. We have also used AUC score to check performance as we can see here auc score for each class is more than 90% that means the probability that the model ranks a random positive point more highly than a random negative point.



Also we can take a look on confusion matrix plot:



Even though the distribution of points in this data set are highly imbalanced still most frequent classes are classified decently though other classes are not having that great performance.

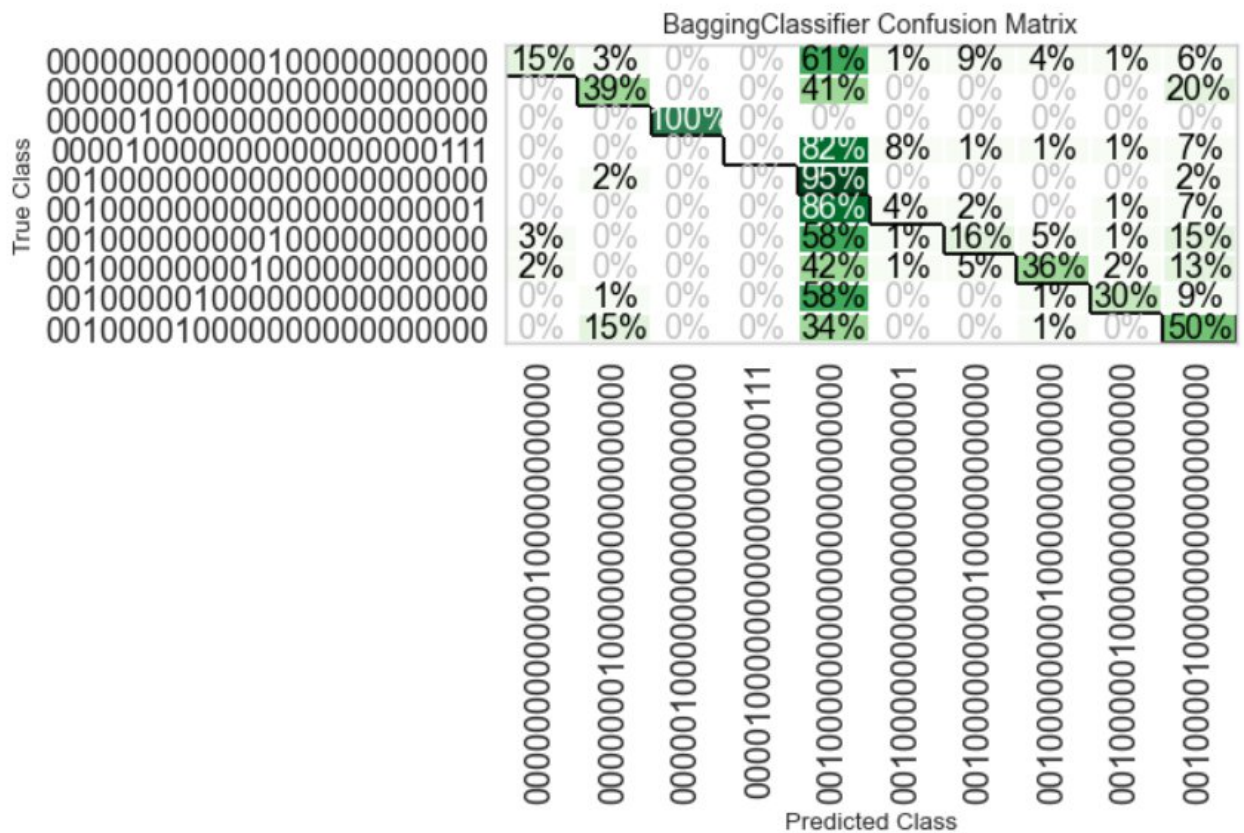
Now after tuning random forest we have got following result:

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7432	0.8690	0.2009	0.6095	0.6670	0.2899	0.3301
1	0.7425	0.8669	0.1949	0.6061	0.6644	0.2734	0.3187
2	0.7390	0.8665	0.1996	0.6062	0.6633	0.2787	0.3171
3	0.7396	0.8657	0.1926	0.6044	0.6621	0.2676	0.3108
4	0.7402	0.8668	0.1979	0.6067	0.6640	0.2794	0.3192
5	0.7383	0.8673	0.1929	0.6049	0.6618	0.2712	0.3112
6	0.7209	0.8694	0.1426	0.5290	0.6068	0.0392	0.1427
7	0.7423	0.8674	0.2019	0.6089	0.6663	0.2885	0.3279
8	0.7416	0.8676	0.2021	0.6092	0.6661	0.2901	0.3280
9	0.7392	0.8651	0.2016	0.6066	0.6638	0.2809	0.3188
Mean	0.7387	0.8672	0.1927	0.5991	0.6586	0.2559	0.3025
SD	0.0062	0.0012	0.0171	0.0234	0.0173	0.0726	0.0536

Hence we are not going to consider random forest for future testings. Now We shall try ensemble on our tuned lightgbm model. After doing that we shall get the following result:

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7720	0.9194	0.3900	0.7284	0.7347	0.4366	0.4530
1	0.7710	0.9185	0.3857	0.7271	0.7332	0.4317	0.4488
2	0.7705	0.9186	0.3898	0.7273	0.7331	0.4317	0.4484
3	0.7717	0.9182	0.3859	0.7269	0.7342	0.4346	0.4514
4	0.7706	0.9193	0.3875	0.7238	0.7326	0.4307	0.4477
5	0.7674	0.9180	0.3839	0.7181	0.7288	0.4226	0.4396
6	0.7726	0.9193	0.3887	0.7279	0.7346	0.4360	0.4532
7	0.7687	0.9185	0.3825	0.7241	0.7299	0.4252	0.4424
8	0.7714	0.9190	0.3834	0.7263	0.7335	0.4338	0.4506
9	0.7710	0.9172	0.3855	0.7289	0.7327	0.4291	0.4472
Mean	0.7707	0.9186	0.3863	0.7259	0.7327	0.4312	0.4482
SD	0.0015	0.0007	0.0025	0.0030	0.0018	0.0043	0.0042

We can easily see the performance of the model is even increased a little bit by doing ensembling. Also taking a look at confusion matrix plot:



As per our best model the ensemble of tuned lgbm is the best model and we shall save it for deployment and use it as our main model for prediction.

This system will be deployed by following steps:

1. First Python script will be made which will contain all necessary codes to create an app user interface and all necessary function to predict using user input data.
2. Also a requirements.txt file needs to be created to define all the dependencies that is required to create this application.
3. Next step is to keep the final_model.pkl to this project folder.
4. create a new git local repository for this project
5. Create a remote to upload and maintain this local repository to [github](https://github.com)
6. Now using [streamlit cloud](https://streamlit.io) to deploy this app hence an account in [streamlit cloud](https://streamlit.io) is required.

7. Inside [streamlit cloud](#) page have to create an app by specifying the [github](#) repository containing this project. Select the proper branch. Click on deploy.
8. App will setup and deploy online after installing all dependencies and fixing all problems in code.

Tools will be used in this project:

- Python
- Git
- Streamlit Cloud (for deploying app)

Libraries will be used in this project:

- [Pycaret](#)
- Streamlit

Python

Python will be used to create the Machine Learning model as well as python will be used to create the app user interface. Python is used as it is very easy to understand and code. Python also have huge set of libraries and built in functions specially for Data Science and Machine Learning. Though python is not among the fastest programming languages and that means more time to process code.

Git

Git is a open source version controlling system. It is used to manage different versions of Projects locally or remotely. Here Git is used for those same reasons. The best feature of Git is it can even track each and every changes of each commit. Different branches can be created and can be changed and used as per user requirements. It can even undo changes and sync it with local repository. Though it has a huge flaw in terms of storage. It can only have 100mb for a single repository. Also more than 25mb can not be uploaded using Git.

[Streamlit Cloud](#)

Streamlit Cloud is a workspace to deploy and collaborate streamlit apps. It is a cloud workspace just like Heroku,aws or azure etc. The best features of Streamlit Cloud is unlimited public apps can be deployed and each app can monthly use upto 1GB bandwidth. Also Streamlit Cloud also allows to deploy a private app with free account. Also there is another feature that is it shows detail logs while the deployment process is going on it is really helpful and user friendly unlike Heroku or aws they only gives the deployment logs but not your app making logs on platform.

[Pycaret](#)

Pycaret is a low code open source machine learning library for python that helps to manage end to end machine learning models. It is very easy to use and saves a lot of time. Best features of Pycaret is very easy to write code and create complex models with just couples of lines of code. It also includes plots and pre and post processing steps.

Streamlit

Streamlit is a python library to create beautiful web apps with very less code and serve interactive feature by default. Streamlit also has performance optimization, data objects saving feature by using cache, Video and audio can be displayed and much more features.

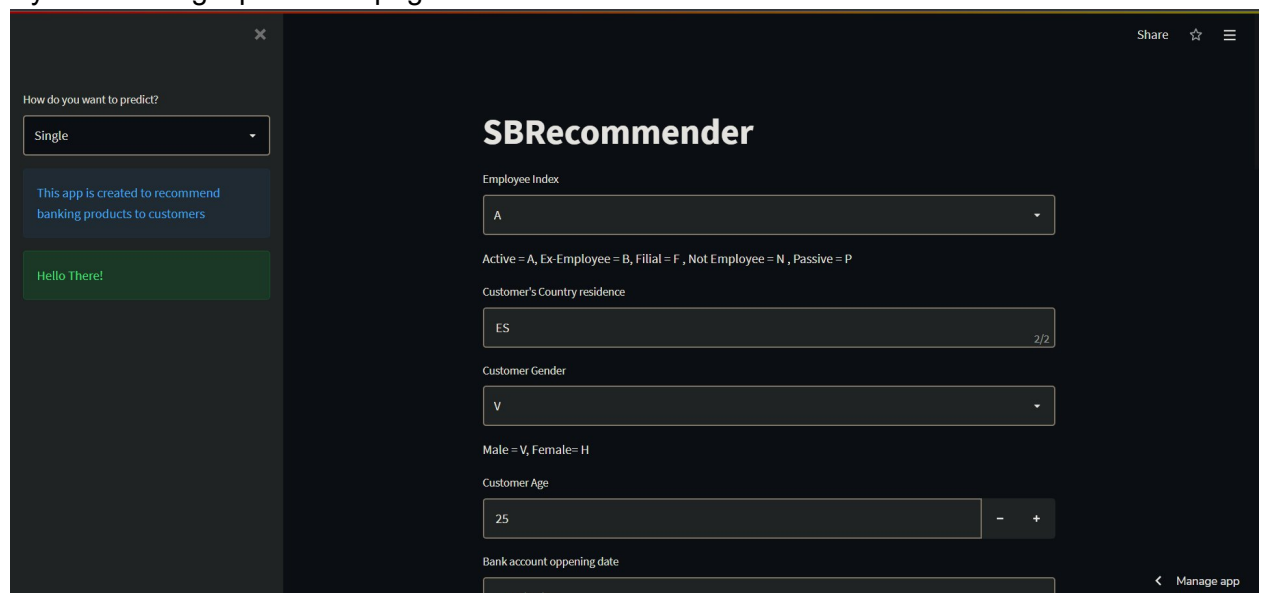
Working demo of the application

Application access url :

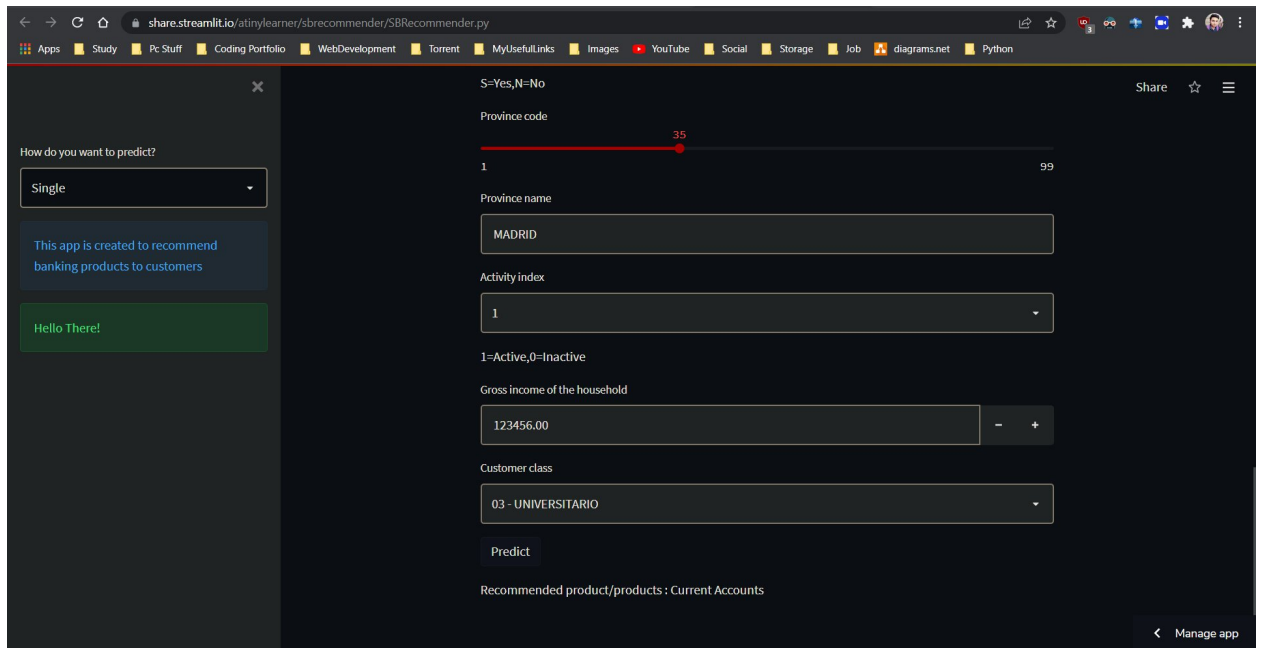
<https://share.streamlit.io/atinylearner/sbrecommender/SBRecommender.py>

It has two primary option for prediction, User can predict a single data by manually inserting that data. Another option is Batch prediction of data where user can upload a csv file with bunch of data and predict all points.

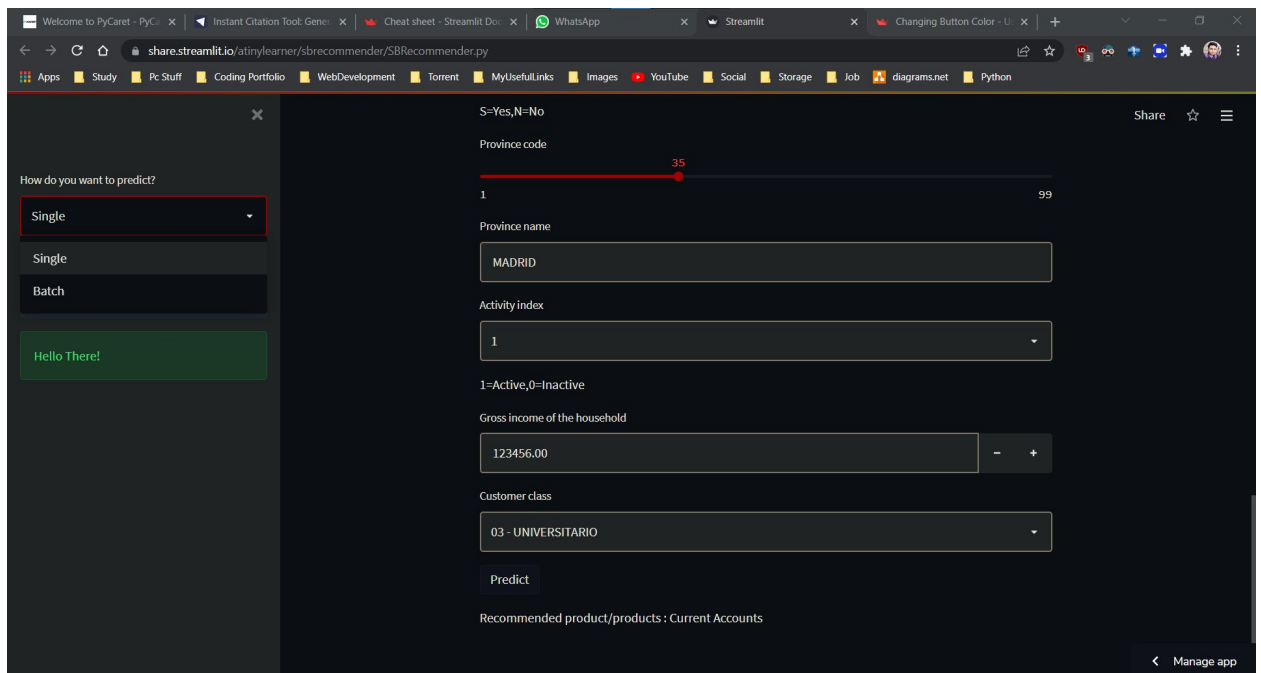
By default Single prediction page starts:



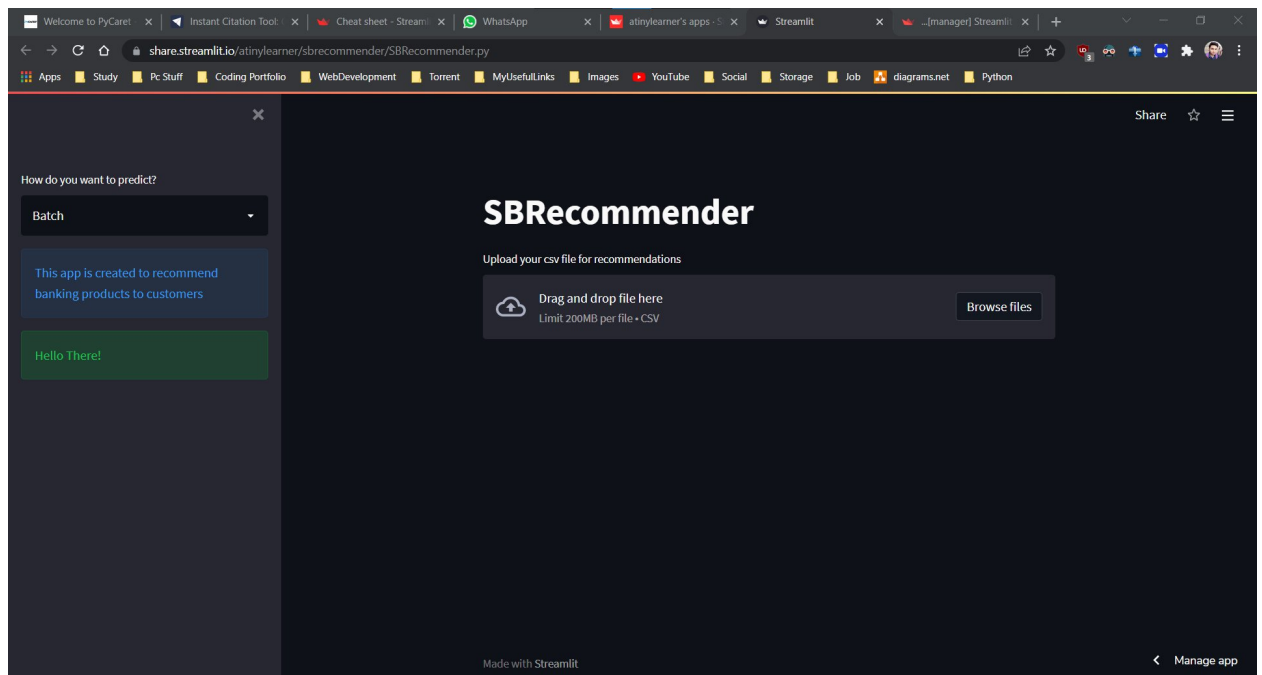
The screenshot shows the SBRecommender web application interface. On the left, a sidebar contains a dropdown menu for 'How do you want to predict?' with 'Single' selected, a description 'This app is created to recommend banking products to customers', and a 'Hello There!' message. The main area is titled 'SBRecommender' and contains several input fields: 'Employee Index' (dropdown with 'A'), 'Active = A, Ex-Employee = B, Filial = F, Not Employee = N, Passive = P', 'Customer's Country residence' (text input with 'ES' and a '2/2' character count), 'Customer Gender' (dropdown with 'V'), 'Male = V, Female = H', 'Customer Age' (text input with '25' and minus/plus buttons), and 'Bank account opening date' (text input with '2019/07/06'). The top right has 'Share', a star icon, and a menu icon. The bottom right has a 'Manage app' link.



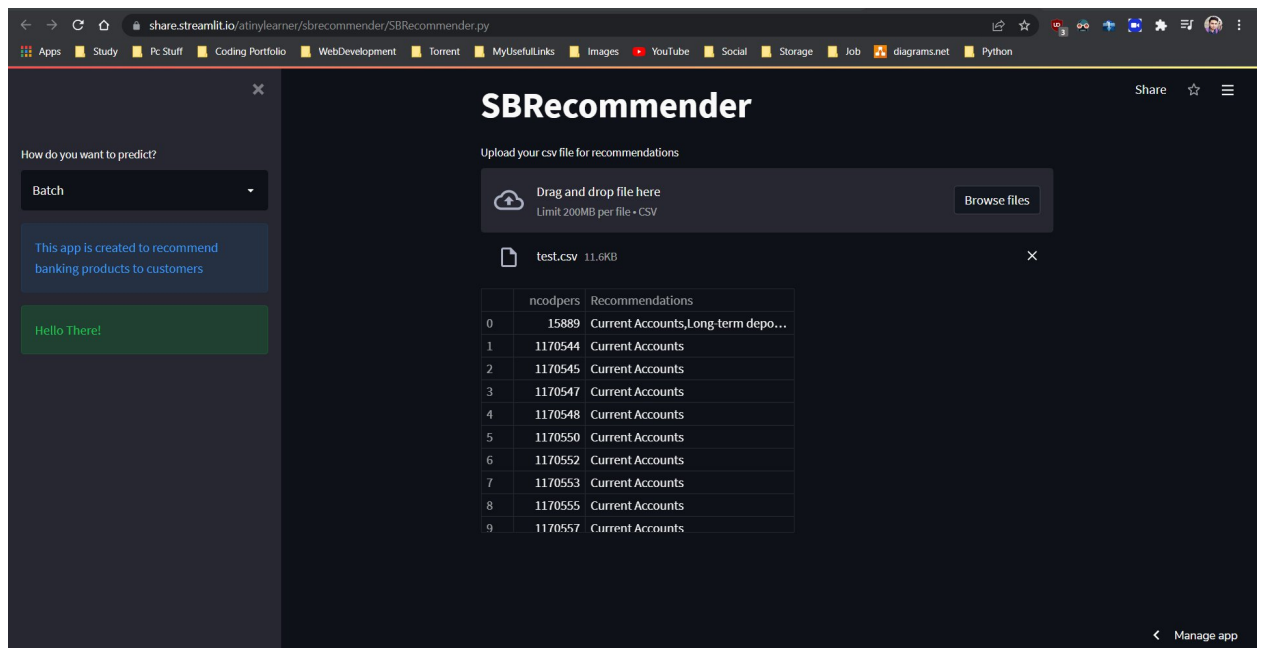
Batch Prediction page can be accessed by clicking on left side panel Select box:



Batch Prediction page looks like below:

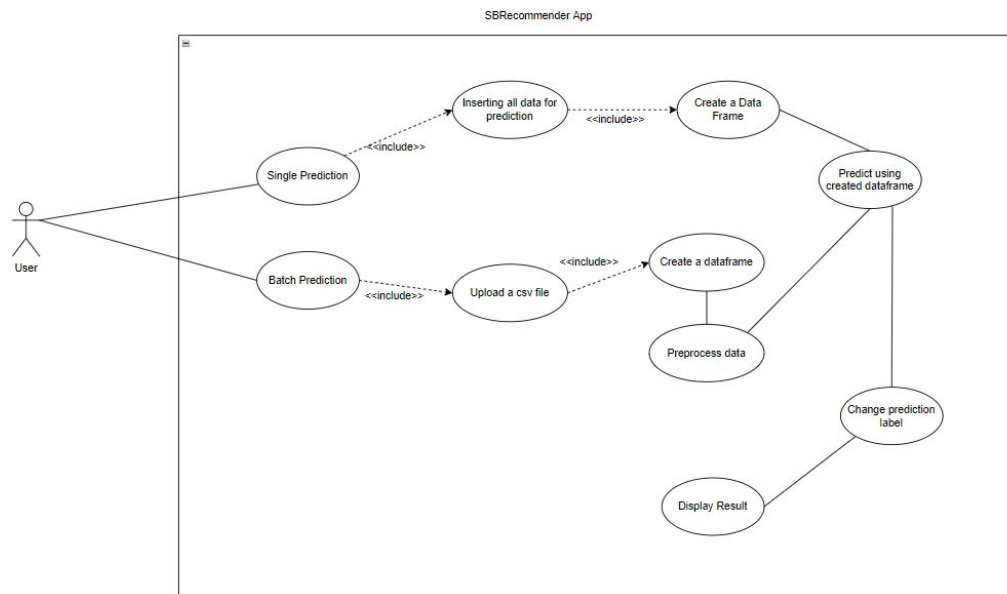


After a file upload it will give predictions:



Architecture Diagrams

Use case diagram for user interactions



Scalability of system

As per storage of the whole project is approximately 20MB. A single prediction will take 1.5sec approximately. Where as data of size 12kb containing 100 data will take almost 1.5 sec i.e similar to a single prediction. Though for larger csv file it will take more time as per csv file size.

The main python script is which runs the app is given below:

```
# All imports
```

```
import datetime
```

```
from pycaret.classification import load_model, predict_model
```

```
import streamlit as st
```

```
import pandas as pd
```

```
import numpy as np
```

```
# loading model
```

```
model = load_model("FM")
```

```
# function to predict results
```

```
def predict(model, input_data):
```

```
    pred_df = predict_model(estimator=model, data=input_data)
```

```
    preds = pred_df['Label'][0]
```

```
    return preds
```

```
# function to change label from 1000000000000000000110011 to Saving Account,Credit  
Card,Securities,Pensions,Direct Debit
```

```
def label_changer(label: str):
```

```
    products = ["Saving Account", "Guarantees",
```

```
                "Current Accounts", "Derivada Account", "Payroll Account", "Junior Account",  
                "Más particular Account", "Particular Account", "Particular Plus Account", "Short-term  
deposits", "Medium-term deposits", "Long-term deposits", "E-Account", "Funds",  
                "Mortgage", "Pensions", "Loans", "Taxes", "Credit Card", "Securities", "Home Account",  
                "Payroll", "Pensions", "Direct Debit"]
```

```
    prod_list = []
```

```
    for i in range(len(label)):
```

```
        if(label[i] == "1"):
```

```
            prod_list.append(products[i])
```

```
    result = ",".join(prod_list)
```

```
    return result
```

```
def preprocess_data(df):
```

```
    # dropping this two columns as they contain all NaN values
```

```
    df = df.drop(columns=['ult_fec_cli_1t', 'conyuemp'])
```

```
    # preprocessing for renta
```

```
    df['renta'] = df['renta'].replace({"NA": '0'})
```

```
    df['renta'] = df['renta'].astype('float')
```

```

df['renta'] = df['renta'].replace({0: np.NaN})

# changing proper type of data for each column

# here fecha_dato and fecha_alta are only having date hence setting them to date
category

df['fecha_dato'] = pd.to_datetime(df['fecha_dato'])
df['fecha_alta'] = pd.to_datetime(df['fecha_alta'])

# Below are all numerical values changing them to proper numerical type

df['age'] = df['age'].astype('int')
df['antiguedad'] = df['antiguedad'].astype('int')

# below columns are categorical data but they already having numeric value

df['indrel'] = df['indrel'].astype('int')
df['ind_nuevo'] = df['ind_nuevo'].astype('int')
df['ind_actividad_cliente'] = df['ind_actividad_cliente'].astype(
    'int')
df['indrel_1mes'] = df['indrel_1mes'].replace(
    {"1.0": "1", "2.0": "2", "3.0": "3", "4.0": "4"})

return df

```

```

def run():

    # creating sidebar with a select box

    add_selectbox = st.sidebar.selectbox(
        "How do you want to predict?", ("Single", "Batch"))

    st.sidebar.info(
        "This app is created to recommend banking products to customers")

    st.sidebar.success("Hello There!")

    st.title("SBRecommender")

    # creating widgets for single pediction page

```



```

if (add_selectbox == "Single"):

    ind_empleado = st.selectbox("Employee Index", options=[
        "A", "B", "F", "N", "P"])

    st.write(
        "Active = A, Ex-Employee = B, Filial = F , Not Employee = N , Passive = P")

    pais_residencia = st.text_input(
        "Customer's Country residence", max_chars=2, value="ES")

    sexo = st.selectbox("Customer Gender", options=["V", "H"])

    st.write(
        "Male = V, Female= H")

    age = st.number_input("Customer Age", min_value=10,
        max_value=120, value=25)

    fecha_alta = st.date_input(
        "Bank account opening date", datetime.date(2019, 7, 6))

    ind_nuevo = st.selectbox(
        "Is customer registered within 6 months?", options=[0, 1])

    st.write(
        "Yes = 1, No= 0")

    antiguedad = st.number_input(
        "Customer seniority(Months)", min_value=1, value=6)

    indrel = st.selectbox("Primary customer?", options=[1, 99])

    st.write(
        "Primary = 1, Not Primary= 99")

    indrel_1mes = st.selectbox("Type of customer", options=[1, 2, 3, 4])

    st.write(
        "1=(First/Primary customer), 2=(co-owner),P=(Potential),3=(former primary),
4=(former co-owner)")

    tiprel_1mes = st.selectbox(
        "Customer relation type", options=["A", "I", "P", "R"])

```

```

st.write(" A=(active), I=(inactive), P=(former customer),R=(Potential)")

indresi = st.selectbox(

    "If the residence country is the same than the bank country?", options=["S", "N"])

st.write("S=Yes,N=No")

indext = st.selectbox(

    "If the customer's birth country is different than the bank country?", options=["S",
"N"])

st.write("S=Yes,N=No")

canal_entrada = st.text_input(

    "Channel used by the customer to join", max_chars=3, value="KHE")

indfall = st.selectbox(

    "User dead or alive?", options=["S", "N"])

st.write("S=Yes,N=No")

tipodom = 1

cod_prov = st.slider(

    "Province code", min_value=1, max_value=99, value=28)

nomprov = st.text_input("Province name", value="MADRID")

ind_actividad_cliente = st.selectbox("Activity index", options=[1, 0])

st.write("1=Active,0=Inactive")

renta = st.number_input("Gross income of the household",

                        min_value=0.00, max_value=30000000.00, step=0.1)

segmento = st.selectbox("Customer class", options=[

    "01 - TOP", "02 - PARTICULARES", "03 - UNIVERSITARIO"])

if(st.button("Predict")):

    input_dict = {'fecha_dato': fecha_alta, 'ncodpers': age*antiguedad, 'ind_empleado':
ind_empleado, 'pais_residencia': pais_residencia, 'sexo': sexo,

                'age': age, 'fecha_alta': fecha_alta, 'ind_nuevo': ind_nuevo, 'antiguedad':
antiguedad, 'indrel': indrel, 'indrel_1mes': indrel_1mes,

                'tiprel_1mes': tiprel_1mes, 'indresi': indresi, 'indext': indext, 'canal_entrada':
canal_entrada, 'indfall': indfall,

```

```

        'tipodom': tipodom, 'cod_prov': cod_prov, 'nomprov': nomprov,
'ind_actividad_cliente': ind_actividad_cliente, 'renta': renta,

        'segmento': segmento}

```

```

input_df = pd.DataFrame([input_dict])

```

```

output = predict(model=model, input_data=input_df)

```

```

result = label_changer(output)

```

```

st.write(f"Recommended product/products : {result}")

```

```

# creating widgets for batch prediction page

```

```

if(add_selectbox == "Batch"):

```

```

    file_upload = st.file_uploader(

```

```

        "Upload your csv file for recommendations", type=["csv"])

```

```

    if(file_upload is not None):

```

```

        data = pd.read_csv(file_upload)

```

```

        final_df = preprocess_data(data)

```

```

        pred_results = predict_model(estimator=model, data=final_df)

```

```

        encode_dict = {}

```

```

        for val in pred_results.Label.unique():

```

```

            encode_dict[val] = label_changer(val)

```

```

        Recommendations = pred_results["Label"].replace(

```

```

            encode_dict)

```

```

        pred_results["Label"] = Recommendations

```

```

        pred_results.rename(

```

```

            columns={"Label": "Recommendations"}, inplace=True)

```

```

        st.write(pred_results[["ncodpers", "Recommendations"]])

```

```

run()

```

Calculating run time complexity:

Loading model function always take same amount of time. It will be any constant value. Predict function will take b unit of time where b represents number of bins which is even smaller than number of data points. Label changer function is having a constant time complexity each time it runs let's say k . Preprocessing will also take a constant time complexity for all points. Let's say each line will take 1 unit of time then total time complexity for preprocessing data will be pp unit of time. Now the run function which is the main function and runs the app in single mode it will take $37+b+k$ units of time will be taken for a single point. Whereas for batch prediction $2+37+b+1+10+4+k*n+pp$, where n is number of points.

Hence total time complexity in worst case is : $54+b+nk+pp$

We can write it as nk time complexity where n is number of points and k is a constant.

Hence this application will take time depending upon the number of points. Hence huge number of data will be predicted but it will take a lot of time.

Though this is not the perfect calculation and system will take even more time for bigger data sets for preprocessing.

Limitations

There are several limitations in this system. First of all no model is perfect. In this machine learning model, it can predict correctly with 76% accuracy and has a precision of 72%. Hence it is not perfect at all. Many improvements can be done to make this model even better.

Due to time and resource limitation I failed to test with various feature engineering technique also various feature interaction is not tested in this model and how those technique will impact here. Tried to test that but due to limited ram system takes a lot of time and then it crashes. Hence failed to test that. Also failed to test throughput and latency. That can be tested and can be improved on using better optimization techniques like dynamic programming and a faster programming language.

Also some more improvements can be done in the user interface of this web app. Various explainable plots can be added to display the result lot more user friendly. User interface can be improved to make the web app more attractive.

The project is hosted on Github: <https://github.com/ATinyLearner/SBRecommender>

Application can be accessed online by using the link:

<https://share.streamlit.io/atinylearner/sbrecommender/SBRecommender.py>

References

Jain,Shubham(August 26, 2017) | Solving Multi Label Classification problems (Case studies included) .Last accessed November 26, 2021:

<https://www.analyticsvidhya.com/blog/2017/08/introduction-to-multi-label-classification/>

Banco Santander(December 14, 2016)| Santander Product Recommendation Can you pair products with people?.Kaggle. Last accessed November 26, 2021:

<https://www.kaggle.com/c/santander-product-recommendation>

Sawtelle,Sonya(October 25, 2016) | Mean Average Precision (MAP) For Recommender Systems. Last accessed November 26, 2021:

<https://sdsawtelle.github.io/blog/output/mean-average-precision-MAP-for-recommender-systems.html>

Taifi,Moussa(November 25,2019) | MRR vs MAP vs NDCG: Rank-Aware Evaluation Metrics And When To Use Them. Last accessed November 26, 2021:

<https://medium.com/swlh/rank-aware-recsys-evaluation-metrics-5191bba16832>

qutbuddin,muffaddal(April 20,2020) | An Exhaustive List of Methods to Evaluate Recommender Systems. Last accessed November 26, 2021:

<https://towardsdatascience.com/an-exhaustive-list-of-methods-to-evaluate-recommender-systems-a70c05e121de>

Khandelwal,Renu(January 6, 2020)| Evaluating performance of an object detection model . Last accessed November 26, 2021:<https://towardsdatascience.com/evaluating-performance-of-an-object-detection-model-137a349c517b>

Shah,Kushi(April 19, 2021)| Exploratory Analysis Using Univariate, Bivariate, and Multivariate Analysis Techniques. Last accessed 26th December 2021:

<https://www.analyticsvidhya.com/blog/2021/04/exploratory-analysis-using-univariate-bivariate-and-multivariate-analysis-techniques/>

<https://www.wikipedia.org/>

[Confusion Matrix for Your Multi-Class Machine Learning Model | by Joydwip Mohajon | Towards Data Science](#)

[Machine Learning Standardization \(Z-Score Normalization\) with Mathematics](#)

<https://pycaret.gitbook.io/docs/get-started/preprocessing>

Ali, M. (2020, April). Pycaret. Retrieved February 27, 2022, from <https://www.pycaret.org>

Streamlit. The Fastest Way to Build and Share Data Apps, <https://docs.streamlit.io/>. Accessed 27 Feb. 2022.