Course Project:

Santander Customer Transaction Prediction

INSY-7120

Mohammad Maydanchi

Introduction:

Kaggle is an online community of data scientists and machine learners, owned by Google. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.[1]

Santander Bank is a wholly owned subsidiary of Spanish Santander Group. It is based in Boston and its principal market is the northeastern United States. It has $57.5 billion in deposits, operates about 650 retail banking offices and over 2,000 ATMs, and employs approximately 9,800 people. It offers an array of financial services and products including retail banking, mortgages, corporate banking, cash management, credit card, capital markets, trust and wealth management, and insurance.[2]

This class project for course INSY-7120 is based on the challenge which Santander asks users of Kaggle to help them identify which customers will make a specific transaction in the future (regardless of the amount of money which transacted). The data that is used for this project has the same structure as real data, We face to around 200000 records and 200 columns for train data and test data, but we do not know the name of variables because of that using intuition and experience based on personal knowledge cannot be helpful.

Approach:

Based on the material of the course I follow Cross Industry Process for Data Mining (CRISP-DM) as my approached step by step.

---

[1] "Kaggle," in *Wikipedia*, April 16, 2019, https://en.wikipedia.org/w/index.php?title=Kaggle&oldid=892701345.
[2] "Santander Bank," in *Wikipedia*, March 24, 2019, https://en.wikipedia.org/w/index.php?title=Santander_Bank&oldid=889223841.

CRISP-DM:

The cross-industry standard process for data mining, known as CRISP-DM, is an open standard process model that describes common approaches used by data mining experts. It is the most widely-used analytics model.[3]

The steps of CRISP-DM are:

1-Business Understanding

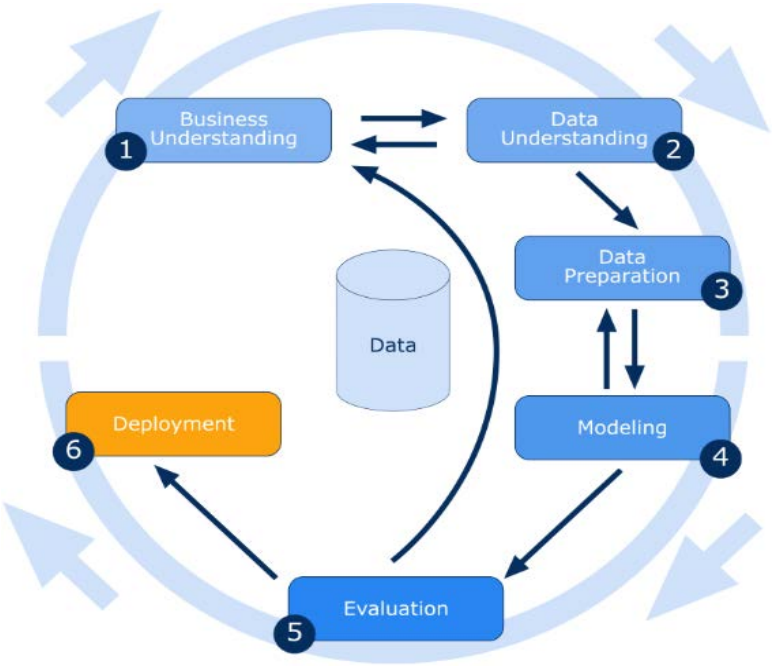
2-Data Understanding

3-Data Preparation

4-Modeling

5-Evaluation

6-Deployment



Step1: Business Understanding

It is crystal clear that this data is related to banking, also there are some features of customers transactions but we cannot identify specific features so the area of our knowledge for business understanding is limited.

[3] "Cross-Industry Standard Process for Data Mining," in *Wikipedia*, April 15, 2019, https://en.wikipedia.org/w/index.php?title=Cross-industry_standard_process_for_data_mining&oldid=892529942.

Step2: Data Understanding:

Both train and test data sets have 200000 rows. Train has 200 numerical variables , target (0,1) and ID_code (string). The test data set does not have a target but has ID_code (string) and 200 numerical variables. Because the test has no target we work just on the train but I do Exploratory Data Analysis(EDA) for both data set and show some features of test data besides the train data set. I have to mention that targets in the train are just 0 and 1 so this problem is categorical.

Train:

	ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	...	var_190	var_191	var_192	var_193	var_194	var_195	var_196
0	train_0	0	8.9255	-6.7863	11.9081	5.0930	11.4607	-9.2834	5.1187	18.6266	...	4.4354	3.9642	3.1364	1.6910	18.5227	-2.3978	7.8784
1	train_1	0	11.5006	-4.1473	13.8588	5.3890	12.3622	7.0433	5.6208	16.5338	...	7.6421	7.7214	2.5837	10.9516	15.4305	2.0339	8.1267
2	train_2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427	14.6155	...	2.9057	9.7905	1.6704	1.6858	21.6042	3.1417	-6.5213
3	train_3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428	14.9250	...	4.4666	4.7433	0.7178	1.4214	23.0347	-1.2706	-2.9275
4	train_4	0	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405	19.2514	...	-1.4905	9.5214	-0.1508	9.1942	13.2876	-1.5121	3.9267

Test:

	ID_code	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	...	var_190	var_191	var_192	var_193	var_194	var_195	var_196
0	test_0	11.0656	7.7798	12.9536	9.4292	11.4327	-2.3805	5.8493	18.2675	2.1337	...	-2.1556	11.8495	-1.4300	2.4508	13.7112	2.4669	4.3654
1	test_1	8.5304	1.2543	11.3047	5.1858	9.1974	-4.0117	6.0196	18.6316	-4.4131	...	10.6165	8.8349	0.9403	10.1282	15.5765	0.4773	-1.4852
2	test_2	5.4827	-10.3581	10.1407	7.0479	10.2628	9.8052	4.8950	20.2537	1.5233	...	-0.7484	10.9935	1.9803	2.1800	12.9813	2.1281	-7.1086
3	test_3	8.5374	-1.3222	12.0220	6.5749	8.8458	3.1744	4.9397	20.5660	3.3755	...	9.5702	9.0766	1.6580	3.5813	15.1874	3.1656	3.9567
4	test_4	11.7058	-0.1327	14.1295	7.7506	9.1035	-8.5848	6.8595	10.6048	2.9890	...	4.2259	9.1723	1.2835	3.3778	19.5542	-0.2860	-5.1612

2-1: Nan Value

There is not any nan in the train data set and test data set.

2-2: Describe

Train:

	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6
count	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000
mean	0.100490	10.679914	-1.627622	10.715192	6.796529	11.078333	-5.065317	5.408949
std	0.300653	3.040051	4.050044	2.640894	2.043319	1.623150	7.863267	0.866607
min	0.000000	0.408400	-15.043400	2.117100	-0.040200	5.074800	-32.562600	2.347300
25%	0.000000	8.453850	-4.740025	8.722475	5.254075	9.883175	-11.200350	4.767700
50%	0.000000	10.524750	-1.608050	10.580000	6.825000	11.108250	-4.833150	5.385100
75%	0.000000	12.758200	1.358625	12.516700	8.324100	12.261125	0.924800	6.003000
max	1.000000	20.315000	10.376800	19.353000	13.188300	16.671400	17.251600	8.447700

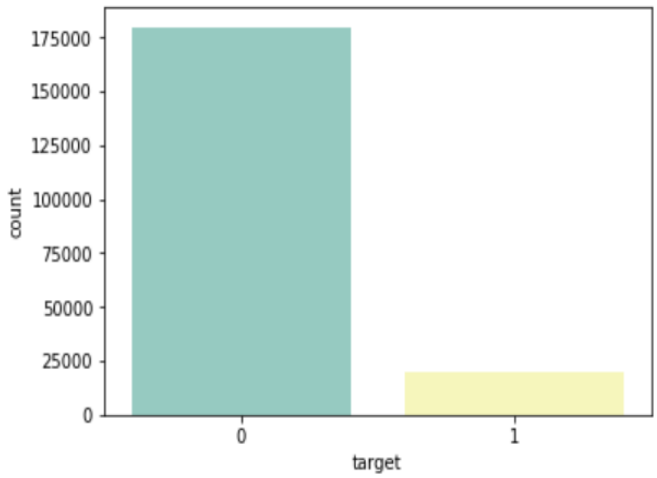
Test:

	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8
count	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000
mean	10.658737	-1.624244	10.707452	6.788214	11.076399	-5.050558	5.415164	16.529143	0.277135
std	3.036716	4.040509	2.633888	2.052724	1.616456	7.869293	0.864686	3.424482	3.333375
min	0.188700	-15.043400	2.355200	-0.022400	5.484400	-27.767000	2.216400	5.713700	-9.956000
25%	8.442975	-4.700125	8.735600	5.230500	9.891075	-11.201400	4.772600	13.933900	-2.303900
50%	10.513800	-1.590500	10.560700	6.822350	11.099750	-4.834100	5.391600	16.422700	0.372000
75%	12.739600	1.343400	12.495025	8.327600	12.253400	0.942575	6.005800	19.094550	2.930025
max	22.323400	9.385100	18.714100	13.142000	16.037100	17.253700	8.302500	28.292800	9.665500

Based on above tables we can say that standard deviation is relatively large for both train and test variable data. It is clear that min, max, mean, standard deviation values for train and test data seem close. And, the mean values are distributed in a large range.

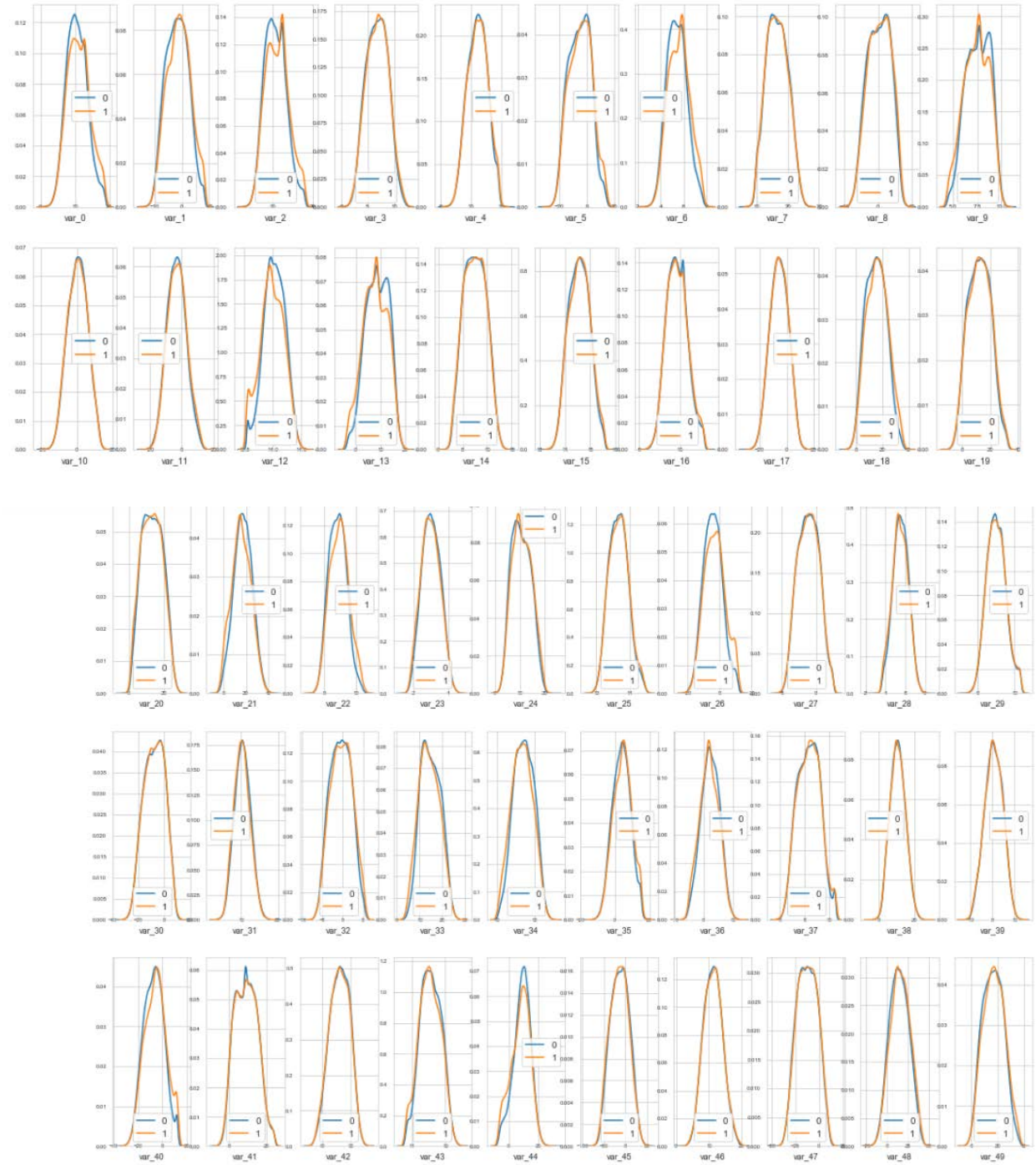
2-3: Distribution of Target

As mentioned before the target in train data is categorical variable. There are 10.049% of target values are 1 and 89.951% are 0.



2-4: Density Plot

We can see density plot for 50 variables in the train data set in below (I did a density plot for all variables in python file):



We can see a lot of variables with considerable different distributions on above plots.

Now we want to check the distribution of each variable on train and test data sets. Below are plots for just 50 variables but I drew plot for all of them in python file:

The train data set and test data set look to be great **balanced**.

2-5: Correlation

One of the most important parts of EDA is checking correlations between variables. If 2 variables have correlation we can ignore 1 of them and reduce the number of variables in the next step of CRISP-DM.

15 least correlated variables:

	level_0	level_1	0
0	var_14	var_46	0.000009
1	var_46	var_14	0.000009
2	var_5	var_39	0.000010
3	var_39	var_5	0.000010
4	var_21	var_29	0.000010
5	var_29	var_21	0.000010
6	var_3	var_1	0.000010
7	var_1	var_3	0.000010
8	var_42	var_15	0.000013
9	var_15	var_42	0.000013
10	var_27	var_10	0.000014
11	var_10	var_27	0.000014
12	var_32	var_43	0.000015
13	var_43	var_32	0.000015
14	var_4	var_37	0.000019

15 most Correlated variables:

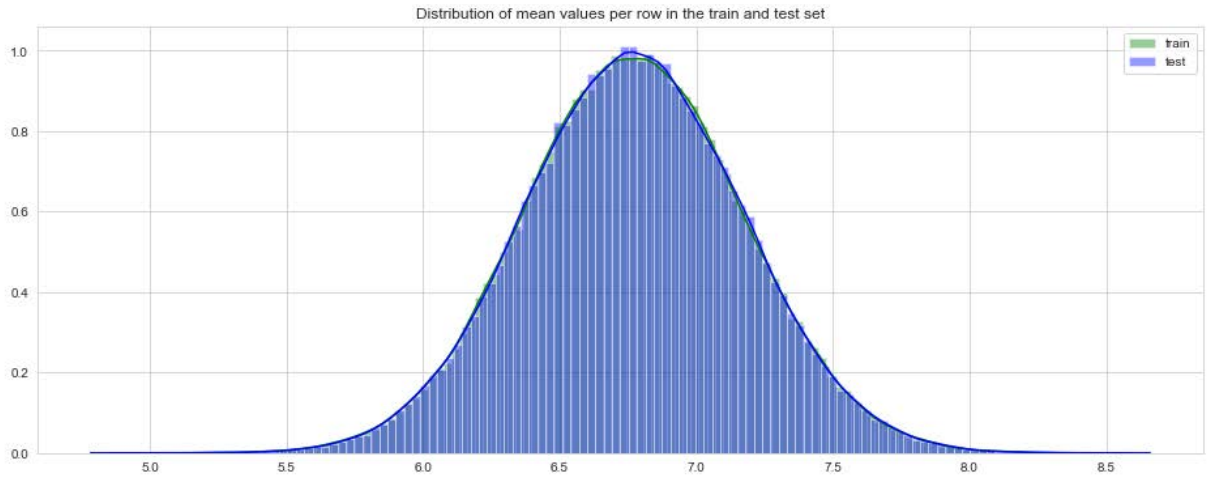
	level_0	level_1	0
2435	var_21	var_2	0.006885
2436	var_32	var_12	0.006936
2437	var_12	var_32	0.006936
2438	var_0	var_6	0.006983
2439	var_6	var_0	0.006983
2440	var_26	var_33	0.006999
2441	var_33	var_26	0.006999
2442	var_23	var_13	0.007298
2443	var_13	var_23	0.007298
2444	var_44	var_13	0.007843
2445	var_13	var_44	0.007843
2446	var_26	var_48	0.007992
2447	var_48	var_26	0.007992
2448	var_2	var_13	0.008795
2449	var_13	var_2	0.008795

As observations show us, the correlation between variables is so small, therefore we can get a conclusion these variables are not correlated at all so we cannot ignore any of them.

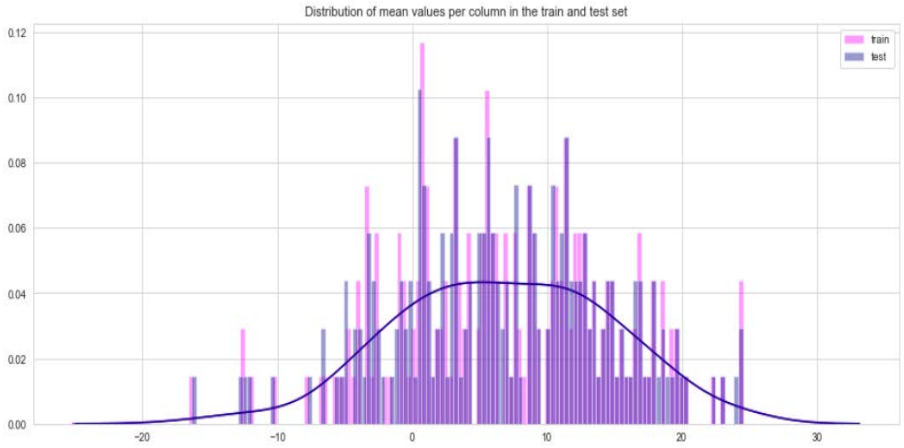
2-6: More Visualizations

For further visualization for each row and column I add 6 more plots:

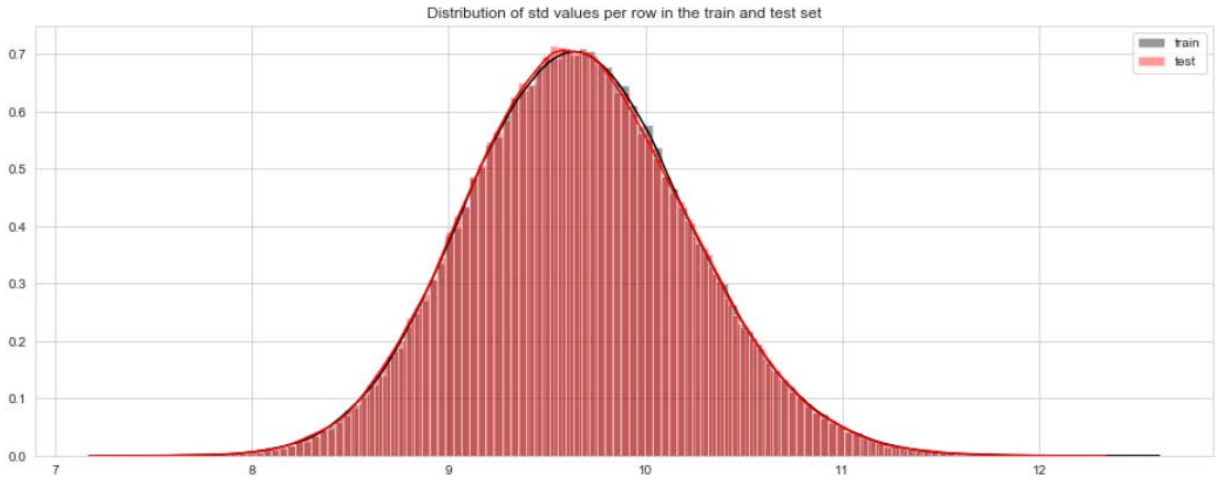
A: Distribution of mean per row in train and test data set:



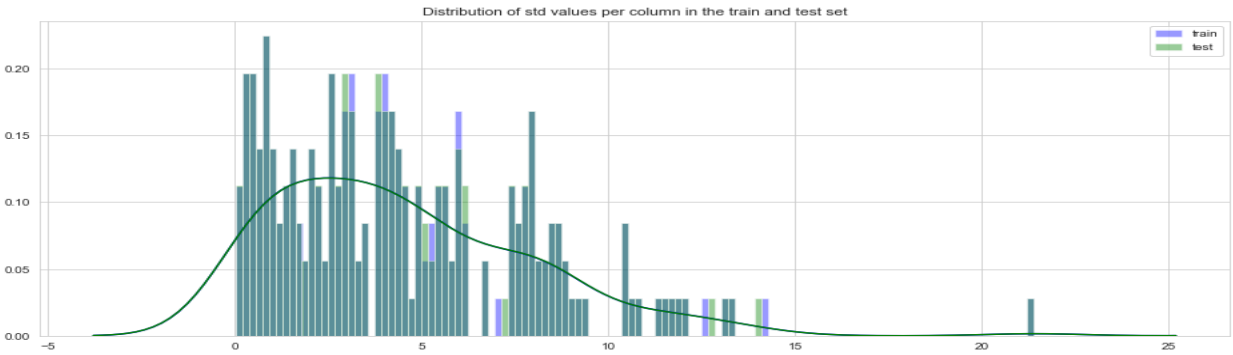
B: Distribution of mean per column in train and test data set:



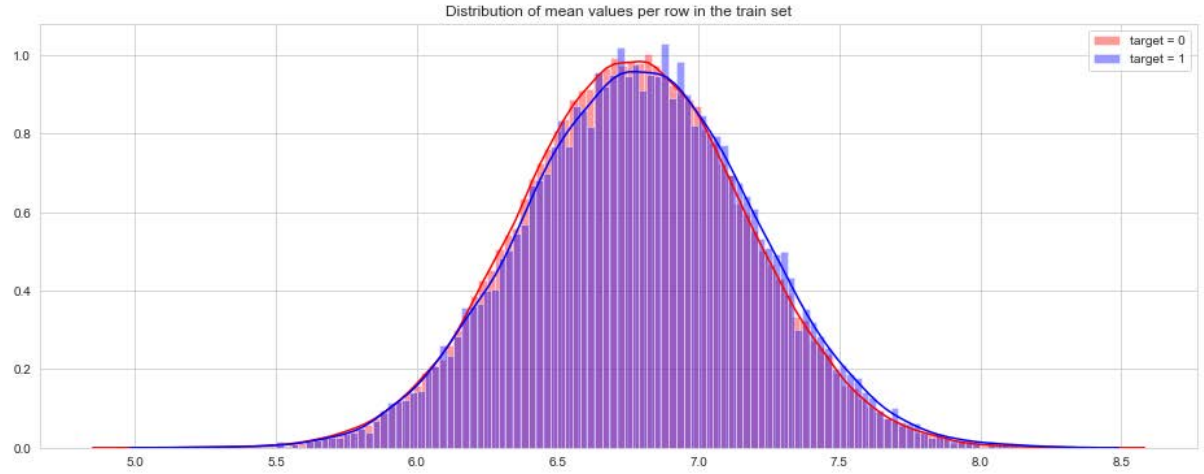
C: Distribution of Standard Deviation of values per row in train and test data set:



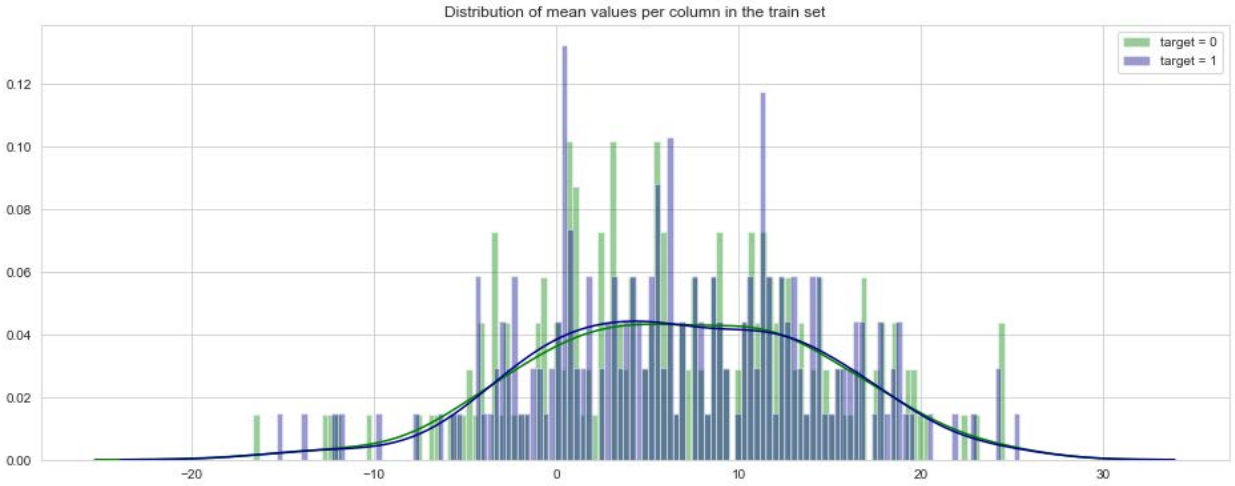
D: Distribution of Standard Deviation of values per column in train and test data set:



E: Distribution of the mean value per row in the training dataset, grouped by the value of the target.



F: Distribution of the mean value per column in the training dataset, grouped by the value of the target.



2-7: Normalization

At the end of the second step, I normalize the data to check the correlation again. For normalization, I use min-max scaling. Also, for making the process simple I drop first column(ID-Code).

	0	1	2	3	4	5	6	7	8	9	...	191
0	0.0	0.427853	0.324824	0.568059	0.388041	0.550670	0.467321	0.454298	0.594255	0.270395	...	0.569515
1	0.0	0.557212	0.428639	0.681235	0.410417	0.628408	0.795072	0.536604	0.500584	0.660911	...	0.668079
2	0.0	0.411969	0.483777	0.578061	0.599690	0.474941	0.471329	0.753295	0.414724	0.270429	...	0.522496
3	0.0	0.535099	0.507140	0.396562	0.546993	0.647586	0.616822	0.572995	0.428577	0.224846	...	0.570474
4	0.0	0.473637	0.533434	0.624133	0.504796	0.621079	0.702836	0.589011	0.622220	0.811883	...	0.387371

Correlation after normalization:

15 least correlated variables:

	level_0	level_1	0
0	var_75	var_191	2.703975e-08
1	var_191	var_75	2.703975e-08
2	var_173	var_6	5.942735e-08
3	var_6	var_173	5.942735e-08
4	var_126	var_109	1.313947e-07
5	var_109	var_126	1.313947e-07
6	var_144	var_27	1.772502e-07
7	var_27	var_144	1.772502e-07
8	var_177	var_100	3.116544e-07
9	var_100	var_177	3.116544e-07
10	var_181	var_12	4.033775e-07
11	var_12	var_181	4.033775e-07
12	var_150	var_116	6.628008e-07
13	var_116	var_150	6.628008e-07
14	var_142	var_44	7.934263e-07

15 most correlated variables:

	level_0	level_1	0
2435	var_21	var_2	0.006885
2436	var_32	var_12	0.006936
2437	var_12	var_32	0.006936
2438	var_0	var_6	0.006983
2439	var_6	var_0	0.006983
2440	var_26	var_33	0.006999
2441	var_33	var_26	0.006999
2442	var_23	var_13	0.007298
2443	var_13	var_23	0.007298
2444	var_44	var_13	0.007843
2445	var_13	var_44	0.007843
2446	var_26	var_48	0.007992
2447	var_48	var_26	0.007992
2448	var_2	var_13	0.008795
2449	var_13	var_2	0.008795

Even after correlation we cannot see any correlation. I think normalization is not necessary for this data so I continue my work with the original train data set.

Step3: Data Preparation:

As mentioned, we do not have any nan values. It is good news because we do not need to consume time for finding and deleting them or replacing value instead of them.

Moreover, we do not have any target in the test data set so I split the train data set into two parts (train, validation). After deleting ID_code, I dedicate 0.7 of data for the train (140000 records and 201 variables) and 0.3 of data for validation (70000 records and 201 variables).

In the other hand, I first did normalization but I as I mentioned in the previous step I do not think it can be helpful for our analyses.

Step4,5: Modeling and Evaluation

First Model: All Variables

Because the target variable is a categorical variable, I use Multiple Logistic Regression. For my first model, I use all variables. Below is a summary of my first model on train data (full results are in R file):

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5955  -0.4015  -0.2326  -0.1236   3.6634

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.816e+01  8.004e+00   7.267 3.67e-13 ***
var_0        5.598e-02  3.616e-03  15.478  < 2e-16 ***
var_1        4.190e-02  2.750e-03  15.236  < 2e-16 ***
var_2        6.140e-02  4.155e-03  14.776  < 2e-16 ***
var_3        1.679e-02  5.469e-03   3.070 0.002138 **
var_4        2.484e-02  6.856e-03   3.623 0.000291 ***
var_5        1.302e-02  1.415e-03   9.206  < 2e-16 ***
var_6        2.711e-01  1.278e-02  21.219  < 2e-16 ***
var_7        2.010e-03  3.264e-03   0.616 0.537928
var_8        1.643e-02  3.355e-03   4.899 9.65e-07 ***
var_9       -1.163e-01  8.974e-03 -12.958  < 2e-16 ***
var_10      -3.711e-03  2.029e-03  -1.829 0.067425 .
var_11       1.256e-02  1.867e-03   6.728 1.72e-11 ***
var_12      -1.121e+00  5.766e-02 -19.439  < 2e-16 ***
var_13      -3.859e-02  2.384e-03 -16.188  < 2e-16 ***
var_14      -1.106e-02  4.968e-03  -2.225 0.026062 *
var_15       1.211e-01  2.708e-02   4.471 7.80e-06 ***
var_16       6.917e-03  4.362e-03   1.586 0.112793
var_17      -9.394e-04  1.665e-03  -0.564 0.572660
var_18       1.729e-02  1.412e-03  12.246  < 2e-16 ***
var_19       4.984e-03  1.390e-03   3.586 0.000335 ***
var_20      -1.028e-02  1.899e-03  -5.411 6.26e-08 ***


    Null deviance: 78279  on 119999  degrees of freedom
Residual deviance: 55580  on 119799  degrees of freedom
AIC: 55982

Number of Fisher Scoring iterations: 6
```

Also, G (deviance(model without predictor)-deviance(model with predictor)) is 26768.84 and p-value is close to zero(computations are in R file). So, we can get a conclusion that our model is significant.

Now we should check our model on validation data and evaluate our model with measures by using Contingency Table:

Contingency Table:

```
   FALSE  TRUE
0 70951  1010
1  5826  2213
```

Evaluation Measures:

```
> Accuracy
[1] 0.9146
> Overal_Error_Rate
[1] 0.0854
> Sensitivity
[1] 0.2689
> specificity
[1] 0.9867
> False_Positive_Rate
[1] 0.0133
> False_Negative_Rate
[1] 0.7311
> Proportion_of_True_Posetive
[1] 0.6933
> Proportion_of_True_Negative
[1] 0.9236
> Proportion_of_False_Posetive
[1] 0.3067
> Proportion_of_False_Negative
[1] 0.0764
```

As you can see the model has very high accuracy. Also, when we compare False Positive Rate and False Negative Rate we see that the model is working better in classify negative records.

This process is fragile because the presence or absence of an outlier in our data can affect our results vastly.

So, for the second mode, I will use K-fold Cross-validation to reduce this effect.

Second Model: Using K-Fold Cross Validation

In the k-fold cross-validation, the original sample is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, a single subsample is retained as the validation data for testing the model, and the remaining *k* − 1 subsamples are used as training data. The cross-validation process is then repeated *k* times, with each of the *k* subsamples used exactly once as the validation data. The *k* results can then be averaged to produce a single estimation. The advantage of this method over repeated random sub-sampling (see below) is that all observations are used for both training and validation, and each observation is used for validation exactly once. [4]

[4] "Cross-Validation (Statistics)," in *Wikipedia*, April 22, 2019, https://en.wikipedia.org/w/index.php?title=Cross-validation_(statistics)&oldid=893640568.

Using this method needs some preparation for data. First, we must identify the target as a factor because our target is categorical variable and for R is not clear that this variable is a categorical variable. Second, we should split the data between train and validation, this time I dedicate 0.6 of my data to train (120000 records and 201 variables) and dedicate 0.4 of my data to validation (80000 records and 201 variables).

These are my results (full results are in R file):

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6721  -0.4004  -0.2324  -0.1237   3.7923

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.829e+01  6.188e+00   9.420  < 2e-16 ***
var_0        5.512e-02  2.799e-03  19.691  < 2e-16 ***
var_1        4.060e-02  2.131e-03  19.053  < 2e-16 ***
var_2        6.562e-02  3.221e-03  20.371  < 2e-16 ***
var_3        1.766e-02  4.240e-03   4.166 3.10e-05 ***
var_4        2.325e-02  5.320e-03   4.370 1.24e-05 ***
var_5        1.382e-02  1.099e-03  12.577  < 2e-16 ***
var_6        2.601e-01  9.878e-03  26.336  < 2e-16 ***
var_7       -1.469e-03  2.533e-03  -0.580 0.561957
var_8        1.811e-02  2.604e-03   6.952 3.60e-12 ***
var_9       -1.116e-01  6.955e-03 -16.045  < 2e-16 ***
var_10      -8.947e-04  1.575e-03  -0.568 0.570101
var_11       1.248e-02  1.446e-03   8.632  < 2e-16 ***
var_12      -1.155e+00  4.479e-02 -25.798  < 2e-16 ***
var_13      -3.888e-02  1.848e-03 -21.045  < 2e-16 ***
var_14      -7.251e-03  3.859e-03  -1.879 0.060224 .
var_15       1.319e-01  2.102e-02   6.276 3.46e-10 ***
var_16       7.982e-03  3.374e-03   2.366 0.017993 *
var_17      -7.083e-06  1.292e-03  -0.005 0.995626
var_18       1.708e-02  1.095e-03  15.593  < 2e-16 ***
var_19       4.341e-03  1.079e-03   4.025 5.70e-05 ***
var_20      -1.081e-02  1.473e-03  -7.340 2.13e-13 ***

    Null deviance: 130463  on 199999  degrees of freedom
Residual deviance:  92380  on 199799  degrees of freedom
AIC: 92782

Number of Fisher Scoring iterations: 6
```

Now we should check our model on validation data and evaluate our model with measures by using Contingency Table:

Contingency Table:

	FALSE	TRUE
0	70951	1010
1	5826	2213

Evaluation Measures:

```
> Accuracy2
[1] 0.9146
> Overal_Error_Rate2
[1] 0.0854
> Sensitivity2
[1] 0.2753
> specificity2
[1] 0.986
> False_Positive_Rate2
[1] 0.014
> False_Negative_Rate2
[1] 0.7247
> Proportion_of_True_Posetive2
[1] 0.6866
> Proportion_of_True_Negative2
[1] 0.9241
> Proportion_of_False_Posetive2
[1] 0.3134
> Proportion_of_False_Negative2
[1] 0.0759
```

It is obvious that the second model as the same as the first model has very high accuracy and it is working better for classifying negative records. All measures are very close to each other for 2 models.

Step 6: Deployment

Based on the Kaggle competition comments for this data it is better to use Magic or LGB. They can get better results for a model. Also, it is better to have a target for test data set, can be very useful because we can test our model with larger data. Moreover, we should know the name of all variables and the importance of them because it can give us an intuition for better analysis specially when we want to prepare our data.

Other Resources:

1- https://github.com/ausim/DataAnalyticsForOperations

2-https://www.youtube.com/watch?v=xkRBfy8_2MU

3-https://www.kaggle.com/gpreda/santander-eda-and-prediction

4-https://thispointer.com/python-pandas-count-number-of-nan-or-missing-values-in-dataframe-also-row-column-wise/

5-https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15

6- https://www.youtube.com/watch?v=OwPQHmiJURI