Santander Customer Transaction Prediction

Overview

At Santander , the mission is to help people and businesses prosper. We are always looking for ways to help our customers understand their financial health and identify which products and services might help them achieve their monetary goals.
Our data science team is continually challenging our machine learning algorithms, working with the global data science community to make sure we can more accurately identify new ways to solve our most common challenge, binary classification problems such as: is a customer satisfied? Will a customer buy this product? Can a customer pay this loan?

Problem Statement

In this challenge, we need to identify which customers will make a specific transaction in the future, irrespective of the amount of money transacted.

Data Sets

We have two types of dataset:

- One is the train data set which contains the target variable.
- Another is test data set which does not contain target variable

Our Goal

We need to predict the target variable in the test dataset with the help of building ML algorithms by using train dataset.
Train data set contains target variable which has two observations

- 0 indicates no which means customer does not make transaction

- 1 indicates yes which means customer makes transactions irrespective to amount

EDA(Exploratory Data Analysis)

Performing EDA is the basic and important step for any Machine Learning or Deep Learning project. It is important because it gives us an overall view of the data and helps us in undercovering the underlying information that might be present in the data. It helps us in the below way:

1. Get an overall view of the data

2. Focus on describing our sample — the actual data we observe — as opposed to making inference about some larger population or prediction about future data to be collected.

3. Identify unusual and extreme cases (outliers, quartile information, etc.)

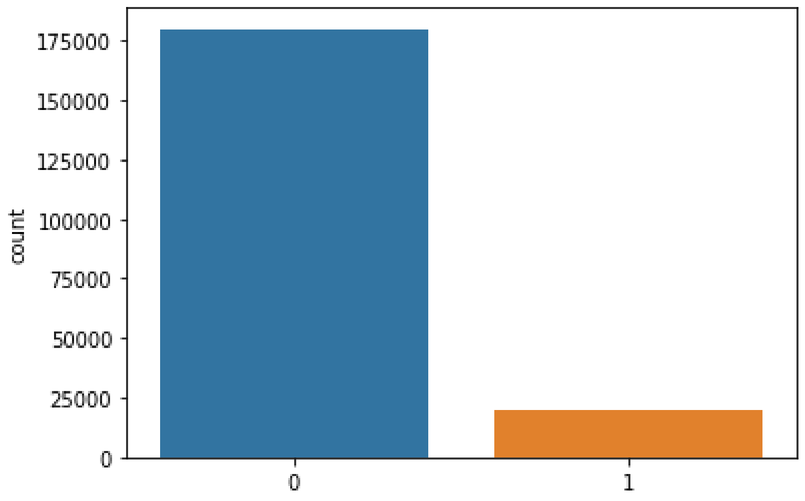
4. Identify the obvious errors in our data which we might have missed otherwise

5. And it itself gives a wide viewpoint of the data and shows us the path on how to proceed with the next steps for any given problem.

In our case we follow the below steps to do the EDA for the problem in our hand:

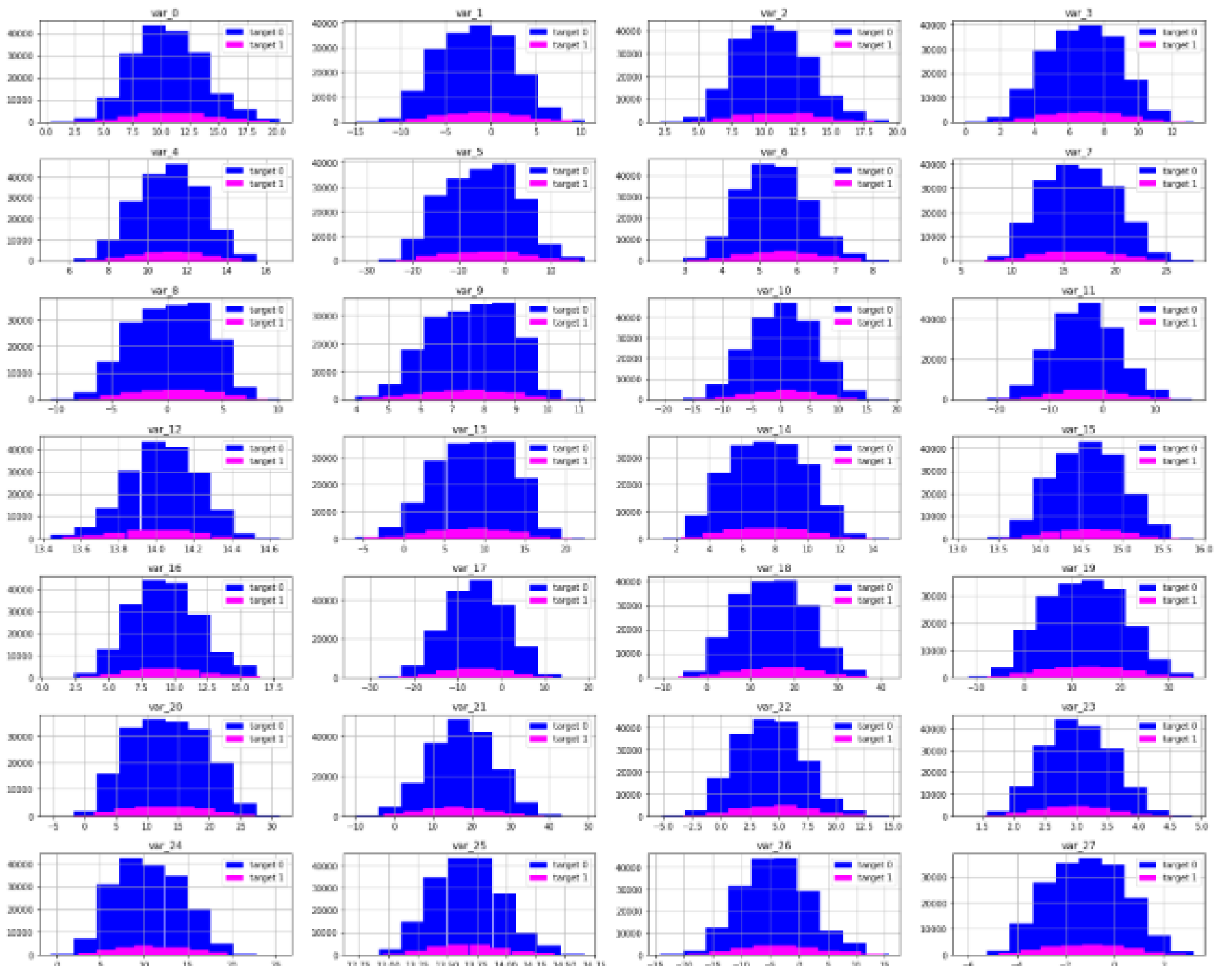
```python
sns.countplot(train['target'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a4300b0208>
```

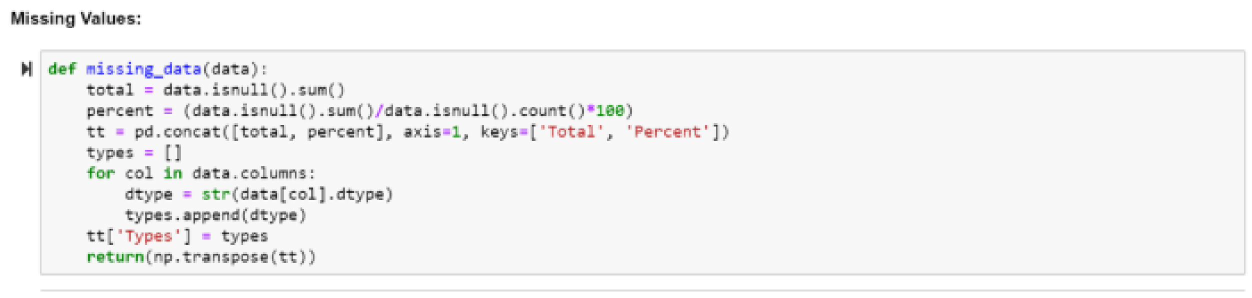


```python
t0=train[train['target']==0]
t1=train[train['target']==1]
```

```python
print('Distributions of 1st 100 features')
plt.figure(figsize=(20,16))
for i, col in enumerate(list(train.columns)[2:30]):
    plt.subplot(7,4,i + 1)
    plt.hist(t0[col],label='target 0',color='blue')
    plt.hist(t1[col],label='target 1',color='magenta')
    plt.title(col)
    plt.grid()
    plt.legend(loc='upper right')
    plt.tight_layout()
```


var_0
var_1
var_2
var_3
var_4
var_5
var_6
var_7
var_8
var_9
var_10
var_11
var_12
var_13
var_14
var_15
var_16
var_17
var_18
var_19
var_20
var_21
var_22
var_23
var_24
var_25
var_26
var_27
target 0
target 1

Finding Missing values:


Missing Values:

```
def missing_data(data):
    total = data.isnull().sum()
    percent = (data.isnull().sum()/data.isnull().count()*100)
    tt = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
    types = []
    for col in data.columns:
        dtype = str(data[col].dtype)
        types.append(dtype)
    tt['Types'] = types
    return(np.transpose(tt))
```

Real life significance of the Problem:

This project can help company in following ways-

1. Segmenting customers into small groups and addressing individual customers based on actual behaviors — instead of hard-coding any preconceived notions or assumptions of what makes customers similar to one another, and instead of only looking at aggregated data which hides important facts about individual customers.

2. Accurately predicting the future behavior of customers (e.g., transaction prediction) using predictive customer behavior modeling techniques — instead of just looking in the rear-view mirror of historical data.

3. Using advanced calculations to determine the customer lifetime value (LTV) of every customer and basing decisions on it — instead of looking only at the short-term revenue that a customer may bring the organization.

4. Knowing, based on objective metrics, exactly what marketing actions to do now, for each customer, in order to maximize the long-term value of every customer.

5. Using marketing machine learning technology that will reveal insights and make recommendations for improving customer marketing that human marketers are unlikely to spot on their own.

Light GBM Model:

LightGBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages:

· Faster training speed and higher efficiency.

· Lower memory usage.

· Better accuracy.

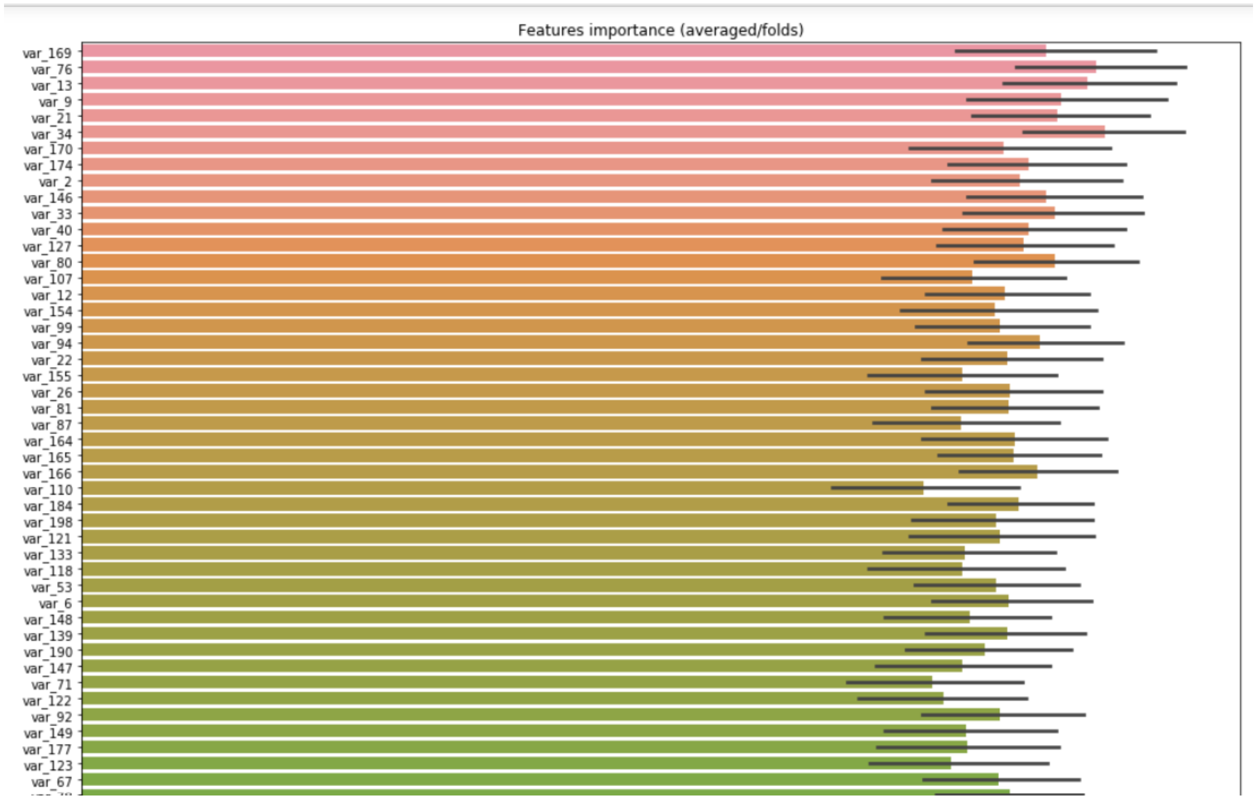
· Support of parallel and GPU learning.

· Capable of handling large-scale data.

Difference from GBDT and Decision Trees:

Light GBM grows trees vertically while other algorithms grow trees horizontally which means that Light GBM grows tree **leaf-wise** while other algorithms grow level-wise. It will choose the leaf with max delta loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm.

Hyperparameters of our Light GBM Model:

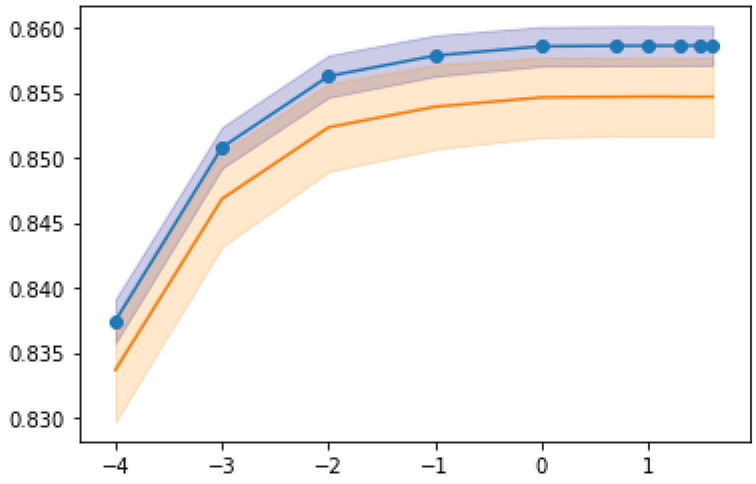


Logistic Regression:

Logistic regression is another technique borrowed by machine learning from the field of statistics, a go-to method for binary classification problems (problems with two class values).

```
Printing parameter Data and Corresponding Log value
==========================================================================
   Parameter value  Corresponding Log Value
0           0.0001                -4.000000
1           0.0010                -3.000000
2           0.0100                -2.000000
3           0.1000                -1.000000
4           1.0000                 0.000000
5           5.0000                 0.698970
6          10.0000                 1.000000
7          20.0000                 1.301030
8          30.0000                 1.477121
9          40.0000                 1.602060

<matplotlib.collections.PathCollection at 0x1ea8f06d320>
```



The logistic function, also called the sigmoid function, was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It’s an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

$$1/(1+e^{-value})$$

Where e is the base of the natural logarithms (Euler’s number or the EXP() function in your spreadsheet) and value is the actual numerical value that you want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.

Representation of user Logistic Regression:

Logistic regression uses an equation as the representation, very much like linear regression.

Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modeled is a binary value (0 or 1) rather than a numeric value.

Below is an example logistic regression equation:

$$y=e^{(b0+b1x)}/(1+e^{(b0+b1x)})$$

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

Conclusion:

We can still improve the model with a bit of hyperparameter tuning of the LGBM model and also we can try some deep learning model so that it can help us in getting a better score.