1003 Customer Transaction Prediction

Team: PTID-CDS-Mar21-1096

Customer Transaction Prediction

Customer transactions could be about anything from opening a checking account to borrowing loans, purchasing a credit card etc.
These actions along with personal details of the customer, the services used, ways in which they interacted with the bank, i.e., online, phone call or a physical visit, etc. help in characterization of customers further leading to implementing targeted strategies and forms one of the bases of customer segmentation.
These information can be used to predict about whether the customer is going to use their service in the future or not.
We as a team have used several models to achieve it.

Problem Statement

To predict if a customer will do a future transaction or not irrespective of the amount of money transacted.

Dataset

```
train.head(5)
```

	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
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
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
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
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
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

5 rows × 202 columns

The dataset is anonymized so we cannot know which feature is what. There is a total of 200 features in this data set along with ID_code and target columns. The target columns contain 0 and 1 value where 0 means the customer will not do a transaction and 1 means the customer will do a transaction.

Tool Kit

Imported Libraries:

- Pandas
- Numpy
- Seaborn
- Matplotlib
- Scipy
- Sklearn.metrics
- Sklearn.model

Method Use:

- EDA
- Feature Engineering
- PCA
- Modelling

EDA (Exploratory Data Analysis)

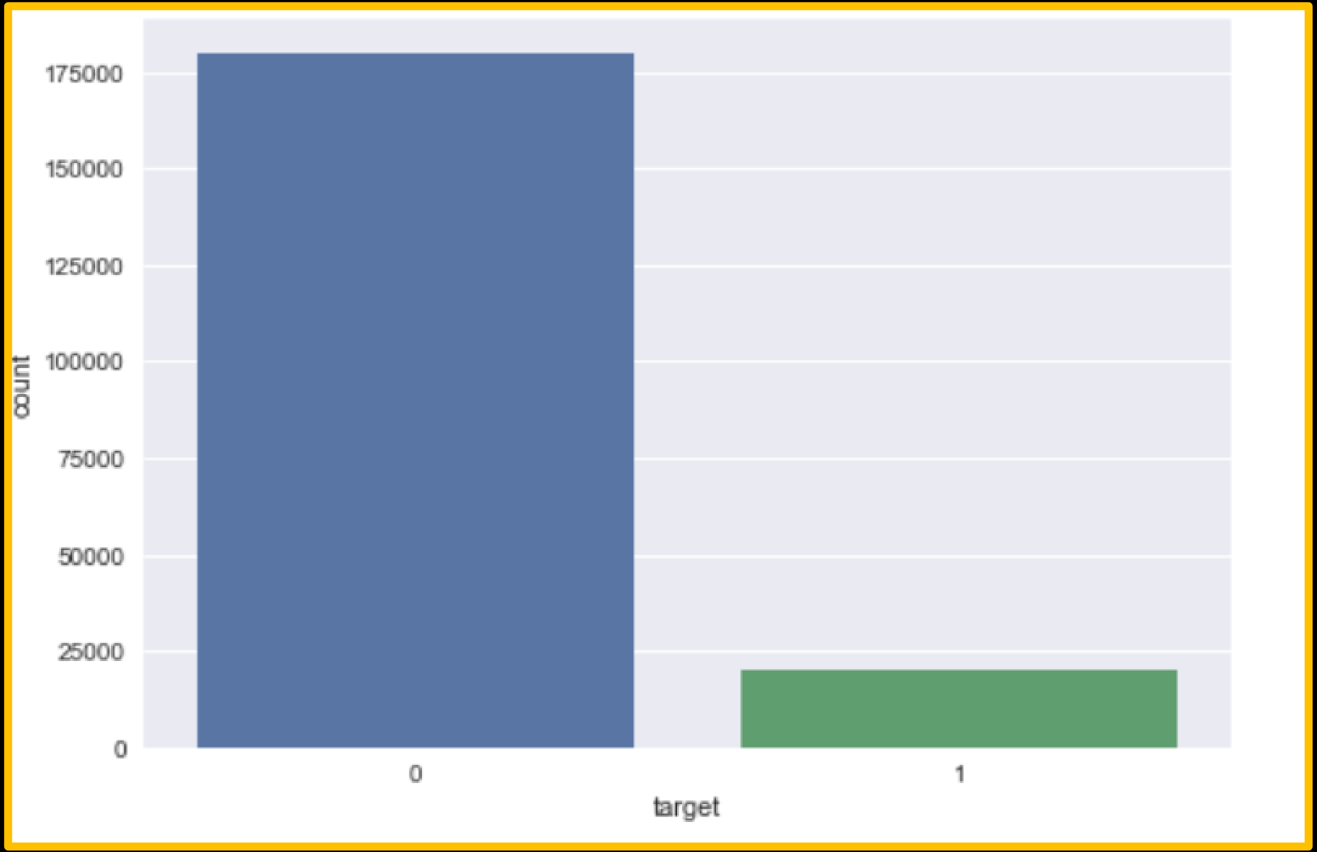
Step 1:
Target Distribution

Step 2:
Variable Distribution

Step 3:
Basics Statistics

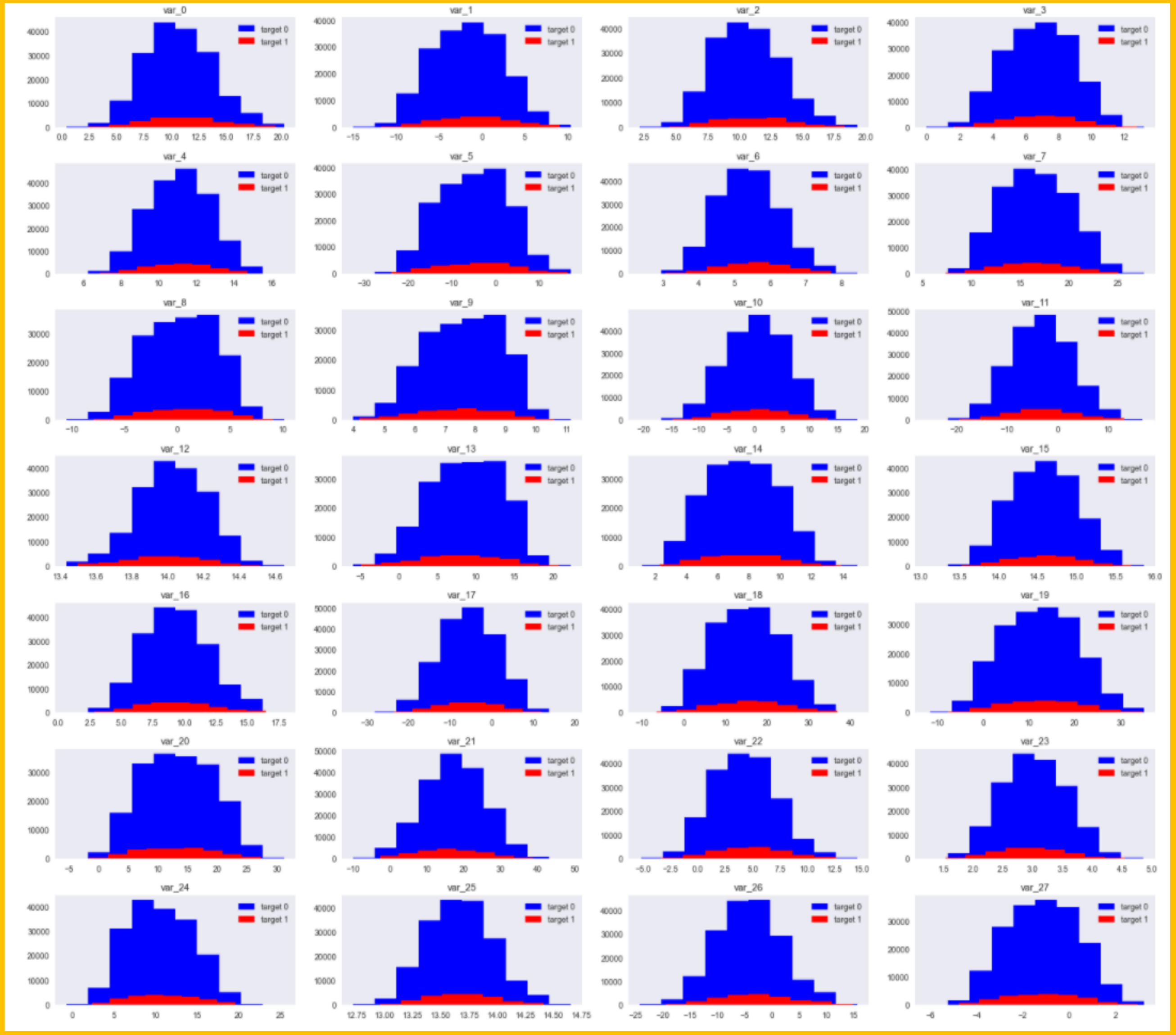
Step 4:
Missing Values

Step 5:
Duplicate Values



Target Distribution

We can see from the above plot that around 90% of our data has 0 (customer did not do a transaction), and around 10% of the data has 1 (customer did do the transaction). This shows that the problem in hand is a binary classification problem and this makes the data very imbalanced.



Variable Distribution

We can see from this that the distribution is nearly Gaussian for all the variables for both the outcome 1 and for outcome 0.

Basic Statistics

Standard deviation is relatively large

In [3]:

```
train.describe()
```

	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	200000.
mean	0.100490	10.679914	-1.627622	10.715192	6.796529	11.078333	-5.065317	5.408949	16.5458
std	0.300653	3.040051	4.050044	2.640894	2.043319	1.623150	7.863267	0.866607	3.41807
min	0.000000	0.408400	-15.043400	2.117100	-0.040200	5.074800	-32.562600	2.347300	5.34970
25%	0.000000	8.453850	-4.740025	8.722475	5.254075	9.883175	-11.200350	4.767700	13.9438
50%	0.000000	10.524750	-1.608050	10.580000	6.825000	11.108250	-4.833150	5.385100	16.4568
75%	0.000000	12.758200	1.358625	12.516700	8.324100	12.261125	0.924800	6.003000	19.1029
max	1.000000	20.315000	10.376800	19.353000	13.188300	16.671400	17.251600	8.447700	27.6918

8 rows × 201 columns

In [23]:
```python
print('train shape:', train.shape)
```

```
train shape: (200000, 202)
```

In [24]:
```python
# Finding missing values in train and test data

def func(df):
    a = df.isnull().sum()
    b = df.count()
    c = (a/b) * 100
    d = pd.DataFrame(a, columns = ['Missingvalue%'])
    return d['Missingvalue%'].sum()
```

In [25]:
```python
print('missing values in train data:', func(train))
```

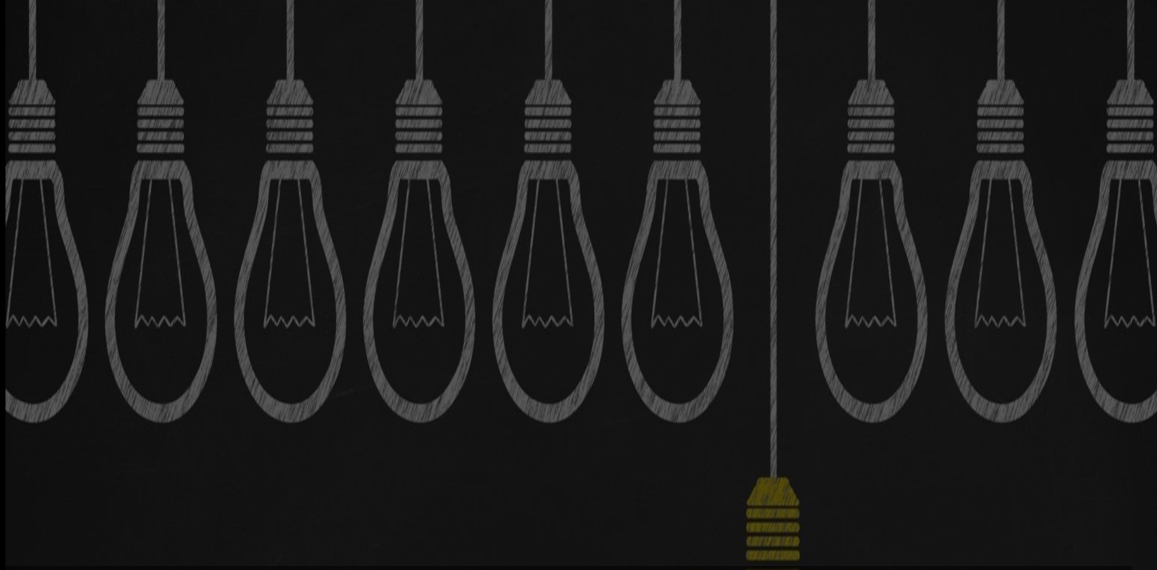
```
missing values in train data: 0
```

Duplicate values

In [18]:
```python
features = train.columns.values[2:202]
unique_max_train = []
for feature in features:
    values = train[feature].value_counts()
    unique_max_train.append([feature, values.max(), values.idxmax()])
```

In [19]:
```python
np.transpose((pd.DataFrame(unique_max_train, columns=['Feature', 'Max duplicates', 'Value'])).\
            sort_values(by = 'Max duplicates', ascending=False).head(15))
```

	66	106	124	10	89	101	146	69	159	23	123	167	41	164	1
Feature	var_68	var_108	var_126	var_12	var_91	var_103	var_148	var_71	var_161	var_25	var_125	var_169	var_43	var_166	var_1
Max duplicates	1084	313	305	203	66	61	59	54	52	41	40	39	39	39	39
Value	5.0214	14.1999	11.5356	13.5545	6.9785	1.6662	4.0456	0.7031	5.7688	13.6723	12.5159	5.6941	11.4522	2.7306	6.863



Missing & Duplicate Values

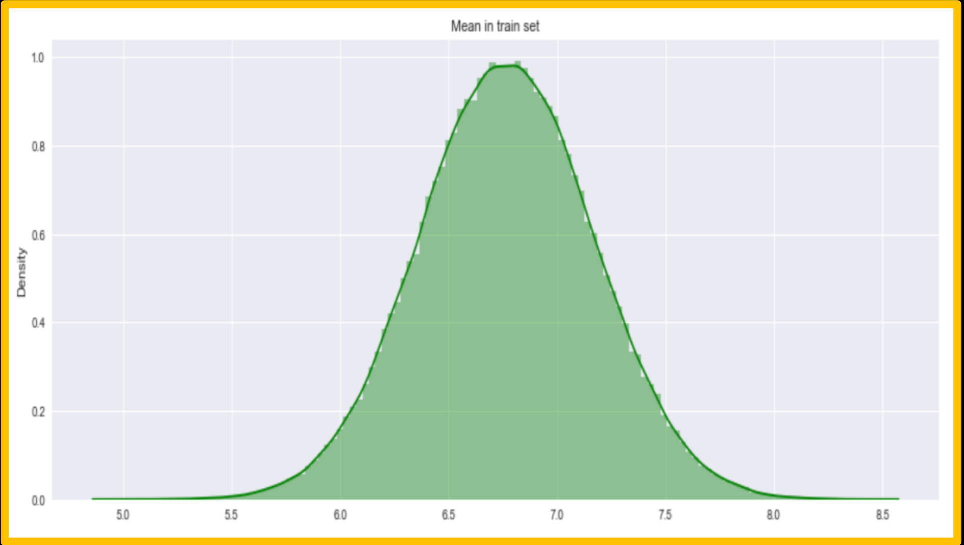
There are no missing values.

We do have some duplicate values which may be used to depict something interesting.

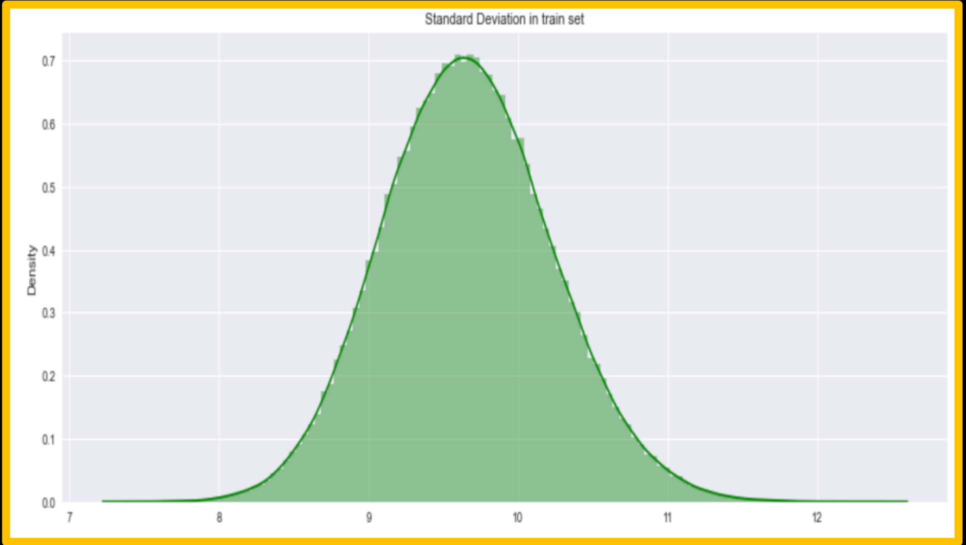
Feature Engineering

- SUM
- MIN
- MAX
- Mean
- Standard Deviation
- Skewness
- Kurtosis
- Median

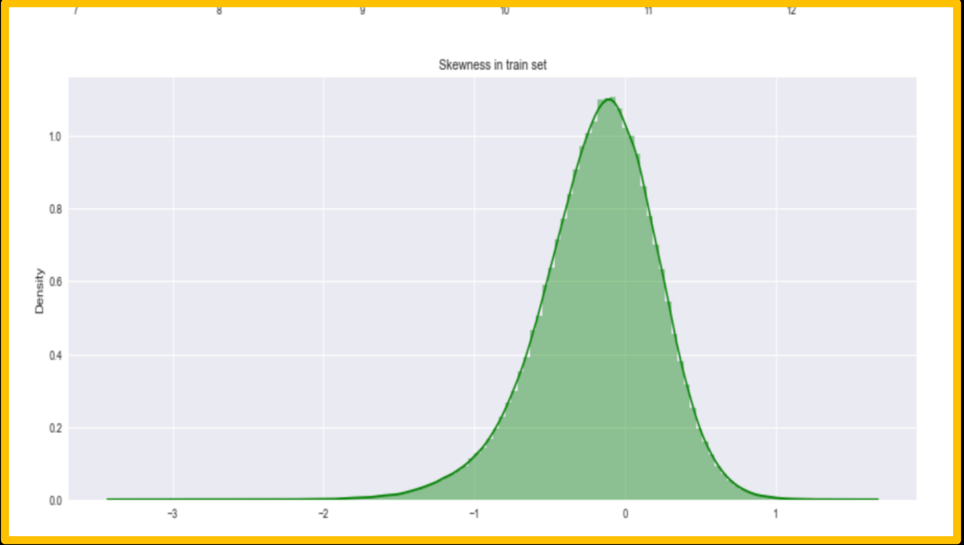
Mean



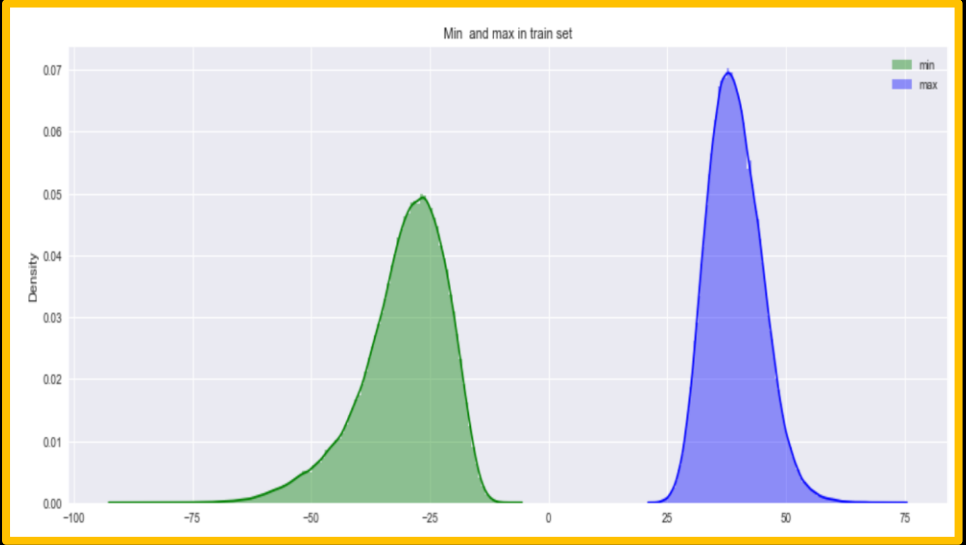
Standard Deviation



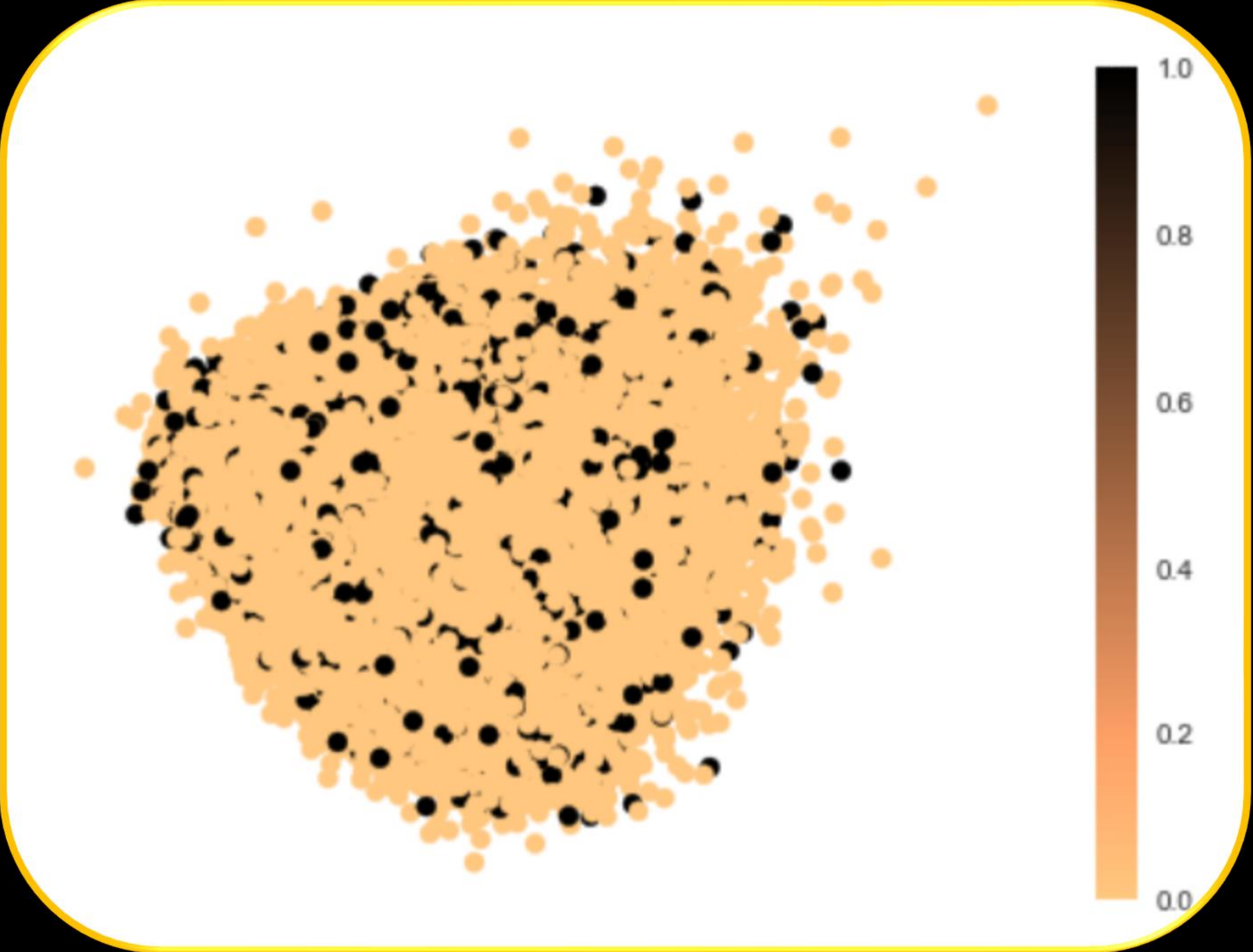
Skewness



Max & Min



We can see from above that the dimension of the data is very high.

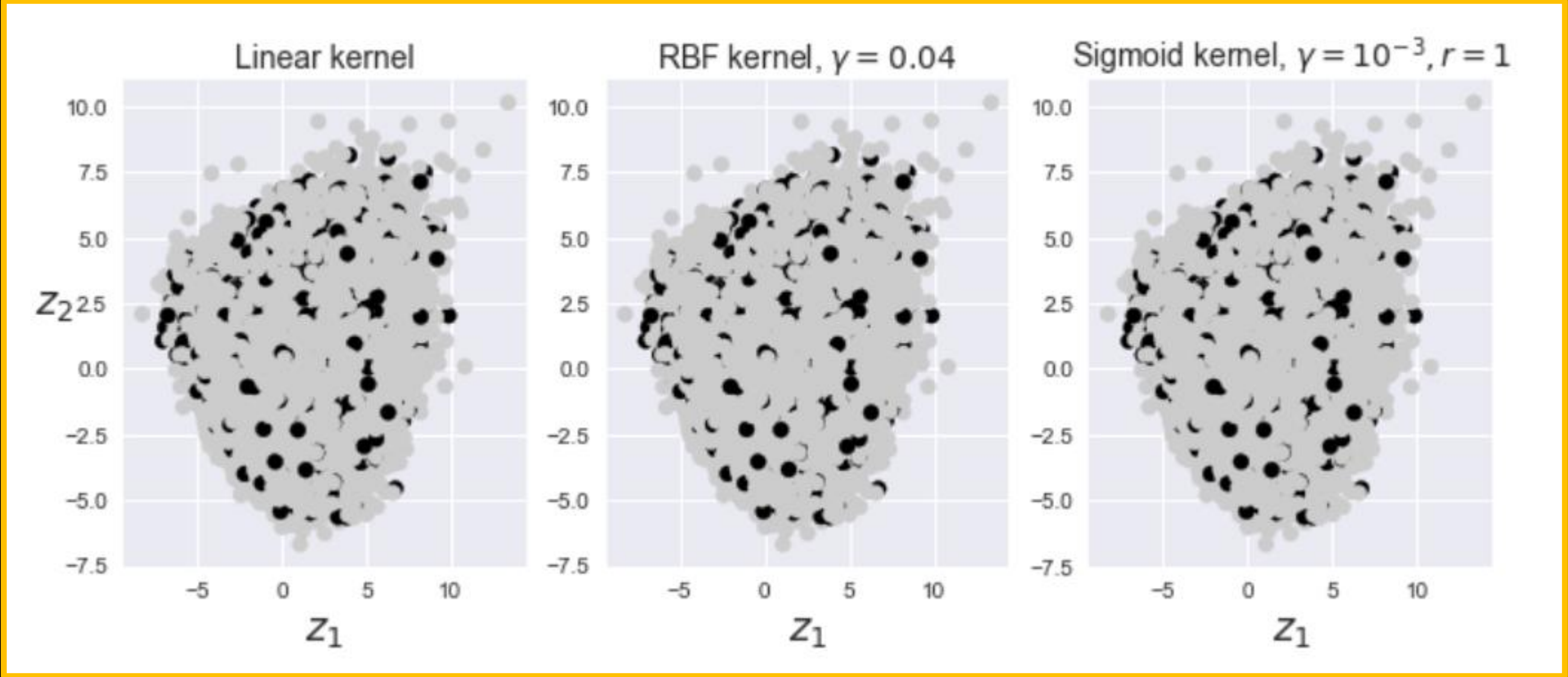


Principal Component Analysis

To check Dimentionality Reduction

As we can see the data cannot be separate. The points are massively overlapped.

Kernel PCA



Since PCA hasn't been useful we decided to proceed with the existing dataset

Models Used

- Linear Regression
- XGBoost
- RandomForest Classifier
- Light GBM
- Kneighbors Classifier

LiGHT GBM

In [33]:
```
import lightgbm as lgb
```

In [34]:
```
param = {
    'bagging_freq': 5,
    'bagging_fraction': 0.335,
    'boost_from_average':'false',
    'boost': 'gbdt',
    'feature_fraction': 0.041,
    'learning_rate': 0.0083,
    'max_depth': -1,
    'metric':'auc',
    'min_data_in_leaf': 80,
    'min_sum_hessian_in_leaf': 10.0,
    'num_leaves': 13,
    'num_threads': 8,
    'tree_learner': 'serial',
    'objective': 'binary',
    'verbosity': -1
}
```

In [35]:
```
num_folds = 11
from sklearn.model_selection import StratifiedKFold,KFold
from sklearn.metrics import roc_auc_score, roc_curve
folds = StratifiedKFold(n_splits= num_folds,random_state=44000)
oof = np.zeros(len(train))
feature_importance_df = pd.DataFrame()

for fold_, (trn_idx, val_idx) in enumerate(folds.split(train.values, target.values)
    X_train, y_train = train.iloc[trn_idx][features], target.iloc[trn_idx]
    X_valid, y_valid = train.iloc[val_idx][features], target.iloc[val_idx]


    print("Fold {}".format(fold_))
```

RandomForestClassifier

In [13]:
```
from sklearn.m
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=15)
```

In [18]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(
    random_state = 20,
    n_estimators=200,
    max_depth=10

)
model.fit(X_train,y_train)
```

```
RandomForestClassifier(max_depth=10, n_estimators=200, random_state=20)
```

Predictions / Evaluate

KNeighborsClassifier

In [35]:
```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=7)
```

In [36]:
```
from sklearn.neighbors import KNeighborsClassifier

model=KNeighborsClassifier(n_neighbors=5)

model.fit(X_train,y_train)
```

```
KNeighborsClassifier()
```

Predictions / Evaluate

In [37]:
```
y_predict = model.predict(X_test)
```

Xgboost

In [29]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=1
```

In [30]:
```
from xgboost import XGBClassifier

model = XGBClassifier(random_state=10, max_depth=5, n_estimators=50)
```

In [31]:
```
model.fit(X_train, y_train)
```

```
C:\Users\surya\Anaconda3\lib\site-packages\xgboost\sklearn.py:888: UserWarning: The use of label encoder in XG
ted and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_labe
onstructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ...
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[20:36:51] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting
default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Exp
c if you'd like to restore the old behavior.


XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=5,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=50, n_jobs=4, num_parallel_tree=1, random_state=10,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

Predictions / Evaluate

In [32]:
```
y_predict = model.predict(X_test)
```

In [33]:
```
from sklearn.metrics import accuracy_score

accuracy_score(y_test, y_predict)
```

```
0.9033501334123926
```

LogisticRegression

In [ ]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=10)
```

In [39]:
```
from sklearn.linear_model import LogisticRegression

model= LogisticRegression(max_iter=1900)
```

In [40]:
```
model.fit(X_train,y_train)
```

```
LogisticRegression(max_iter=1900)
```

Predictions / Evaluate

In [41]:
```
y_predict = model.predict(X_test)
y_predict
```

```
array([0, 1, 0, ..., 0, 0, 0], dtype=int64)
```

Code Snippets

	Linear Regression	RandomForest Classifier	XGBoost	Kneighbours Classifier	Light GBM
Why	It's the most simple modelling method and the go-to method for any data scientist.	Random forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time.	Xgboost can automatically handle missing data values and learn the best direction for missing values. The performance and execution speed is also good.	This is used for both regression as well as for classification but mostly it is used for Classification problems. Since this seemed like a Classification problem this method seemed wise to use.	Our dataset is imbalanced and thus we should not use traditional models like Logistic regression. We should assure independence of predictor variables and we almost have uncorrelated predictor variables.
Train_ Accuracy	0.91	0.90	0.96	0.90	0.95

CHALLENGES AS TEAM

- Coordination across 3 time zones
- Coordination among new people
- Who will do what and the work division based on strength and weakness.
- Balance office work and completion of project.

THANK YOU