Machine Learning Engineer Nanodegree

Capstone Project

Predict which products will an Instacart consumer purchase again

Vinay K

August 31th, 2019

1. Definition

Project Overview

Instacart is an online grocery delivery service provider. The main objective of this project is to use data on customer orders over time to predict which previously purchased products will be in a user's next order.

The dataset consists of relational set of files describing customer's orders over time. The dataset is anonymised and contains a sample of over 3 million grocery orders from more than 200,000 Instacart users. For each user there is between 4 and 100 of their orders, with the sequence of products purchased in each order.

The entity that are provided includes:

- Aisles
- Departments
- Order_Products_Prior
- Order_Products_Train
- Orders
- Products

Datasets are self-explanatory where **Aisles** dataset represent on which aisle product was placed, **Departments** dataset represent to which department product belongs and **Products** dataset represent item that customers buy.

Orders dataset includes data related to when each order is placed and an **ID** being referred in Order_Products_Prior and Order_Products_Train**,** which uniquely identifies what products are bought in each order.

Problem Statement

Same day delivery service for online orders are becoming more common, in order to meet the consumer demand there is a need for better understanding of:

- Where the customer are located
- From which store orders are being placed
- What items are being bought
- How many items are repeated in subsequent orders
- Frequency of orders

Original input dataset is transformed to correlate relationship between product and consumer's buying pattern. This transformed dataset is used as input to our model.

Inputs to our model will be feature dataset extracted along with label specifying whether product is reordered or not - thus this can be seen as classification problem having 2 labels and output tells probability of product being ordered again for future orders.

Metrics

Since our dataset is not balanced we will be using F-Score as evaluation metrics, F score is the Harmonic mean between precision and recall. The greater the F score, the better is the performance of our model, which is defines as follows

```
F = ((1+beta**2)*TruePositive) / (((1+beta**2)*TruePositive) +
beta**2*FalseNegative+FalsePositive
```

Where,

True Positive = correctly predicted the product in next order

True Negatives = correctly predicted the product not in next order

False Positive = wrongly predicted the product in next order

False Negative = failed to predict the product in next order

2. Analysis

Data Exploration

a. Orders:

	order_id	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_prior_order
0	2539329	1	prior	1	2	8	NaN
1	2398795	1	prior	2	3	7	15.0
2	473747	1	prior	3	3	12	21.0
3	2254736	1	prior	4	4	7	29.0
4	431534	1	prior	5	4	15	28.0

eval_set tells whether given data is used for training, testing or preparing features prior to training.

days_since_prior_order tells days since last order.

b. Order_products_prior / train:

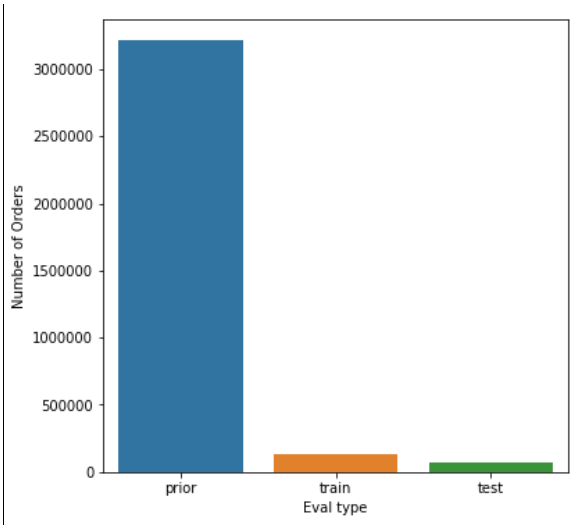
	order_id	product_id	add_to_cart_order	reordered
0	2	33120	1	1
1	2	28985	2	1
2	2	9327	3	0
3	2	45918	4	1
4	2	30035	5	0

reordered tells whether that particular product is ordered before from that customer.

Below we can check **descriptive Stats** of extracted features along with visualization.

Exploratory Visualization and Descriptive Statistics

a. Count orders type



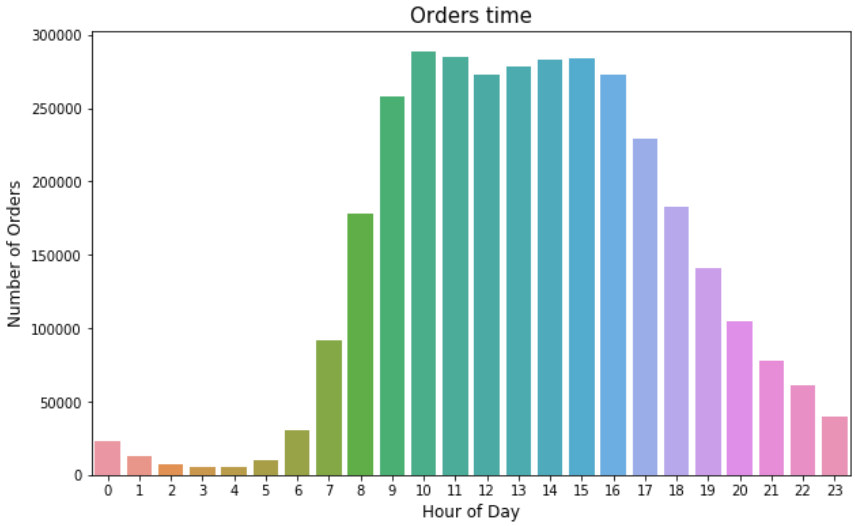
Prior:	3214874
Train:	131209
Test:	75000

In orders dataset we have 3.2m prior, 131k train and 75k test records.

b. At what time customers are making orders

```
order_hour_count.describe()
```

```
count        24.00000
mean     142545.12500
std      114125.98611
min        5474.00000
25%       28586.25000
50%      122430.50000
75%      272625.00000
max      288418.00000
```

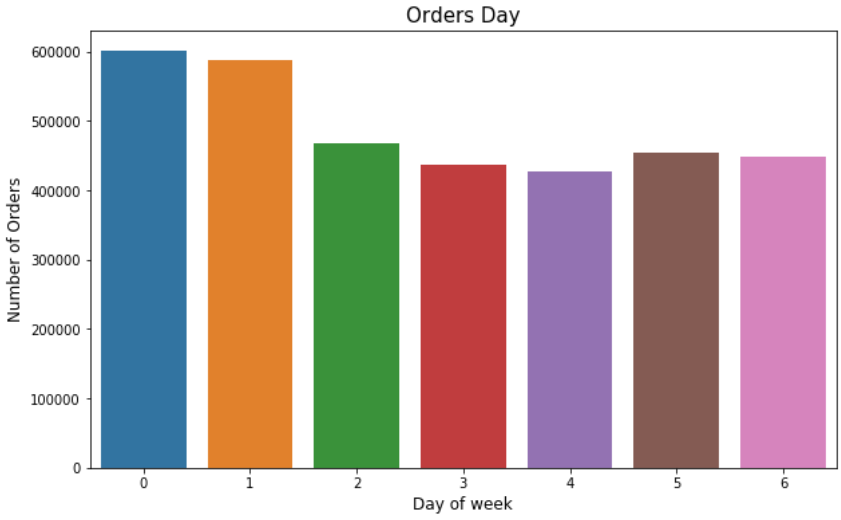


Inference: Looking at the graph it's clear that most orders were made during 7 am and 8 pm.

c. On what day of a week customers are making orders

```
order_day.describe()
```

```
count         7.000000
mean     488726.142857
std       73274.459369
min      426339.000000
25%      442866.500000
50%      453368.000000
75%      527369.000000
max      600905.000000
```

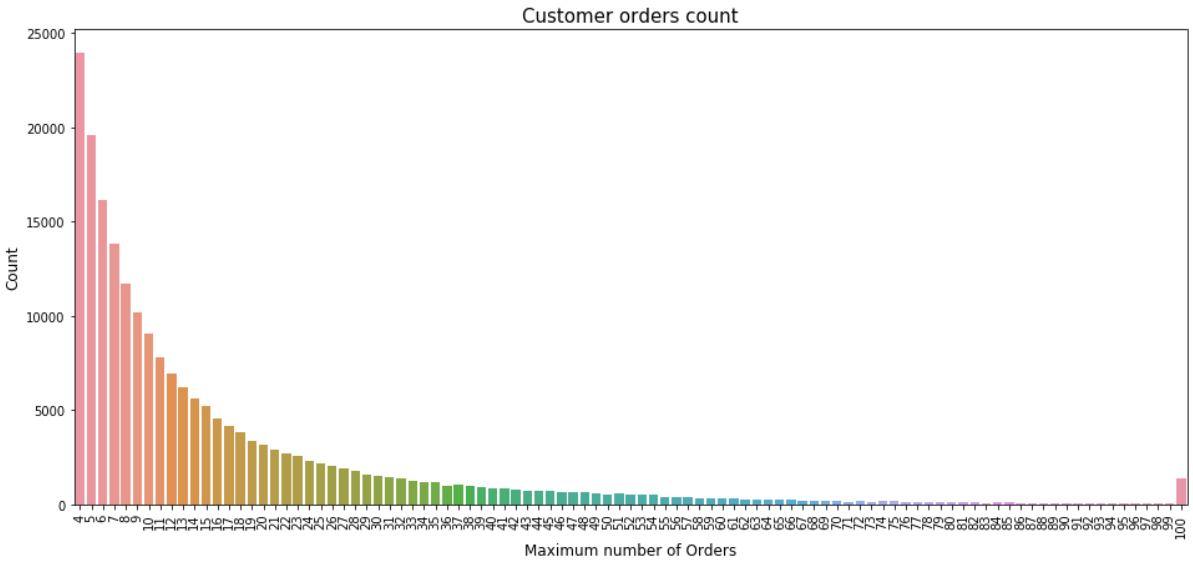


Inference: Assuming week starts on Sunday we can tell customers are buying relatively more on Sunday and Monday.

d. Number of orders grouped by count

```
split_orders_count.describe()
```

```
count       97.000000
mean      2125.865979
std       4144.374897
min         47.000000
25%        138.000000
50%        554.000000
75%       1779.000000
max      23986.000000
```



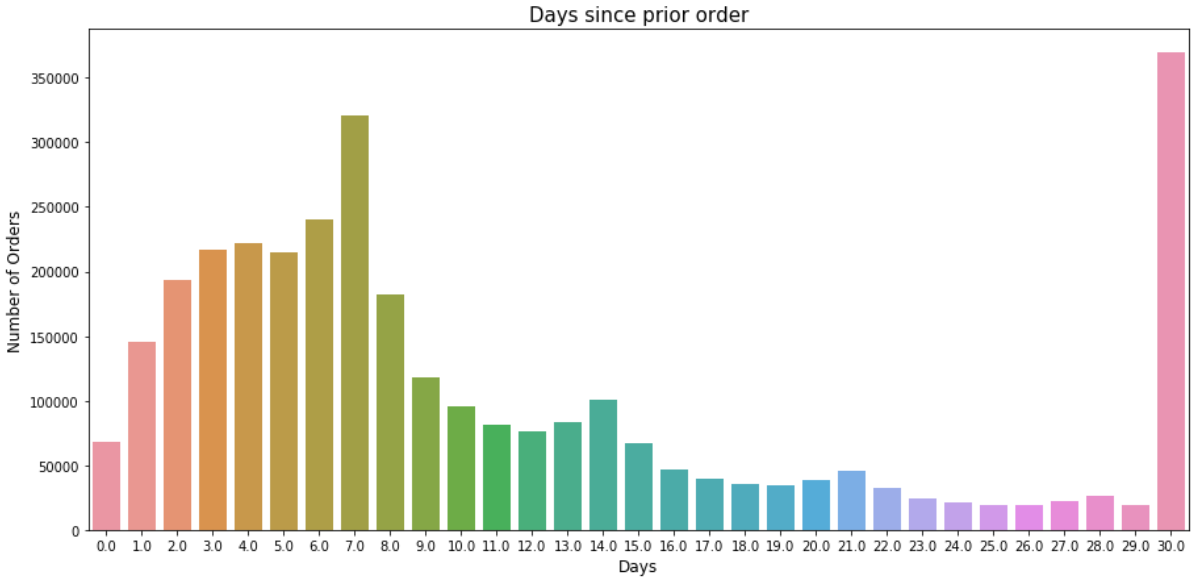
Inference: It indicates distribution orders count from all customers

- 24,000 orders were by customers who had total order count 4
- 5,000 orders were by customers who had total order count 15

e. When the next order is placed

```
days_since_prior_order.describe()
```

```
count        31.000000
mean     103705.612903
std       95317.188052
min       19016.000000
25%       33198.000000
50%       67755.000000
75%      163482.000000
max      369323.000000
```

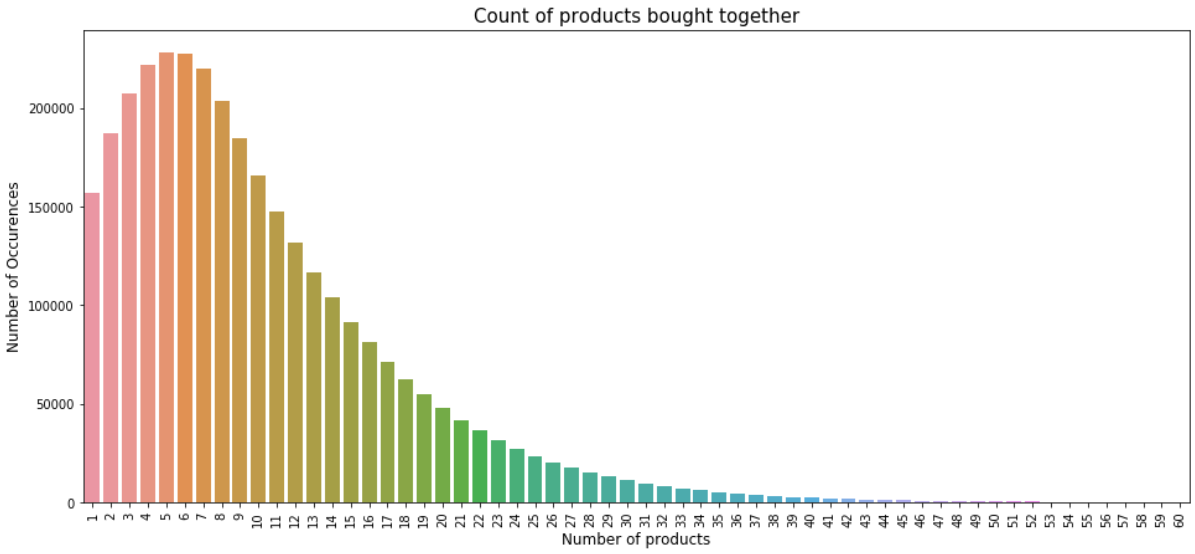


Inference: More customers are ordering again between day 1 – day 15 and on 30th day from their last order date.

f. Number of products in each order

```
order_products_cnt.describe()
```

```
count        60.000000
mean      53567.683333
std       75285.846549
min         119.000000
25%        1009.750000
50%       10411.000000
75%       83805.000000
max      228330.000000
```



Inference: most orders have products in the range of 1 to 20.

Algorithms and Techniques

We will use neural network to find solution to our problem, particularly deep neural network which has more than one hidden layer. On each hidden layer, the neural network learns new feature space by computing transformations of given inputs and then apply non-linear function which in turn will be input of the next layer. The process continues until we reach the output layer.

Coding the Neural Network:

- **Forward Propagation:** we use the network with its current parameters to compute a prediction for each example in our training dataset. We use the known correct answer that a human provided to determine if the network made a correct prediction or not. An incorrect prediction, which we refer to as a prediction error, will be used to teach the network to change the weights of its connections to avoid making prediction errors in the future.
- **Backward Propagation:** we use the prediction error that we computed in the last step to properly update the weights of the connections between each neuron to help the network make better future prediction. We use a technique called **gradient descent** to help us decide whether to increase or decrease each individual connection's weights, Training Rate is used to determine how much to increase or decrease the weights during each training step. We repeat this process for each training sample in the training dataset, and then we repeat the whole process many times until the weights of the network become stable.

We will use TensorFlow implementation of Neural Network for our model as it performs optimizations that speed up our code.

Benchmark

Since the problem statement is taken from kaggle competition, we will use the Leaderboard and the winners score as benchmark.

F-Score of 0.40972 on Public Leaderboard is observed as best score for the above said problem statement.

Ref: https://www.kaggle.com/c/instacart-market-basket-analysis/leaderboard

3. Methodology

Data Pre-processing

We are ignoring Aisles dataset as it doesn't influence customers buying habit and using remaining dataset we will create new feature set for training along with labels. Feature Engineering is necessary as raw data is not in right format and features are selected in such a way that they are the most useful as well as relevant from the available data.

We will do many transformation to extract information which is hidden in dataset:

a. **Basket size** – orders dataset doesn't contain any info on how many products are bought in that respective order instead order_products_prior has a field 'add_to_cart_order', using which we can combine both dataset and extract number of products in order which is basically the maximum value of 'add_to_cart_order'.

b. **Products Ordered** – since there are two dataset one containing order and other products in order, we need to merge these two and create a single dataset.
c. **Mean** – save the mean of basket size, time and days since order was placed. This is one type of feature scaling representing standardisation of independent variables in the features of a dataset.
d. **Distinct Products** – find how many distinct products are ordered by a customer.
e. **Repeated Products** – find how many products are repeated overall.
f. Merge the fields we extracted into original feature_dataset we will be referencing in future.
g. **Min-Max Normalisation** – it represents rescaling of the data features, used to bring all values in the range of [–1,1]
h. Now we have feature_dataset but not the training labels which is necessary to train our model for making future prediction, using 'eval_set' == 'train' we can prepare products in next order by each customer that acts as labels.
i. **Data Splitting** – split the feature_dataset and labels for training, test_trained and test. The training set represent the known and labelled data which is used while building a machine learning model, this set of data helps in predicting the outcome of the future data. The test set represents the set of data used to test the predicted hypothesis and test_trained set is used to validate or retest the created hypothesis in order to avoid overfitting.

Implementation

a. We began our project by stating our goal, that is to predict whether a given product will be ordered or not by a given customer.
b. Next step in our implementation is to clean the data – given dataset doesn't contain any duplicates or missing values.
c. Next step is feature extraction, which build values intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps.
d. Neural network with following architecture was used.

```
np.random.seed(9)

model = Sequential()
model.add(Dense(16, input_dim=len(feature_list), activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

- Model employs 5 hidden layers
- Adam optimization algorithm combines ideas from RMSProp and Momentum.
- Accuracy metric is used to judge the performance of our model.

e. Train

```
# fit the model
model.fit(X_train, Y_train, epochs=10, callbacks=callbacks_list)

scores = model.evaluate(X_train, Y_train, verbose=0)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

Epoch: a number of epochs means how many times you go through your training set.

The model is updated each time a batch is processed, which means that it can be updated multiple times during one epoch.

Refinement

We made several decisions in our training. For instance, the number of layers, the sizes of the layers, the number of epochs, etc.

Size of input layer	Number of hidden layers	Size of hidden layers	Loss Function	Optimizer	Accuracy	F-Score
16	3	64	binary_crossentropy	adam	90.79	0.39132
16	5	64	binary_crossentropy	adam	90.80	0.39165
16	6	64	binary_crossentropy	adam	90.81	0.39124
32	6	64	binary_crossentropy	adam	90.83	0.39055
16	7	64	binary_crossentropy	adam	**90.80**	**0.39292**
64	7	128 – 5 layers 256 – 2 layers	binary_crossentropy	adam	90.79	0.38939
16	7	32 – 3 layers 64 – 3 layers 128 – 1 layer	binary_crossentropy	adam	90.79	0.39141
16	9	64	binary_crossentropy	adam	90.79	0.39177
16	9	32 – 3 layers 64 – 3 layers 128 – 3 layers	binary_crossentropy	adam	90.79	0.38765

Since we have over 8 million dataset, training over entire dataset for selecting best model takes more time, so above we restricted our train_set to subset of input dataset. Once we have best performing model we can again train the model over large train_set.

Model which gave F-Score of 0.39292 was selected with respected hyperparameters for further training.

4. Results

Model Evaluation and Validation

In order to check the robustness of our model, we need to validate our model against unseen data. Overfitting is suspect when the model accuracy is high with respect to the data used in training the

model but drops significantly with new data. Effectively the model knows the training data well but does not generalize. This makes the model useless for purposes such as prediction.

Simple way to detect this effect in practice is cross validation. This attempts to examine the trained model with new data set to check its predictive accuracy. We used different hyperparameters to arrive at best model over sample dataset, now we will train the model with 65% of total dataset which constitutes as train_set and use rest 35% as validation_set.

For 3 million orders we will extract labels from existing dataset which will be used to validate against the result of model prediction.

Accuracy = (True Positive + True Negative) / Total Number of Samples

```
True Positive: 130154
True Negative: 2466080
False Positive: 238793
False Negative: 164973
```

Accuracy = (130154 + 2466080) / (130154 + 2466080 + 238793 + 164973) = 0.86

86% accuracy on unseen data tells us there is no overfitting as training accuracy is **90%** and thus no significant variance is observed.

Accuracy looks at only correct prediction but in reality false positive and false negative miss-classification have adverse effect. So below we will justify our validation methodology.

Justification

F-Score is a measure of test's accuracy in statistical analysis of binary classification, it considers both the precision and recall of the test to compute the score.

F-Score of our model is **0.39198** and F-Score of benchmark model is **0.40972**. Since F-Score provide a more realistic measure of test's performance so we can confidently say our final model and solution is significant enough to have adequately solved the problem.

5. Conclusion

Confusion Matrix: We can measure the effectiveness of our model through the confusion matrix. It is a performance measurement for machine learning classification.

(True Positive) correctly predicted the product in next order	**(False Positive)** wrongly predicted the product in next order
(False Negative) failed to predict the product in next order	**(True Negative)** correctly predicted the product not in next order

Below are the values obtained from our f-score function:

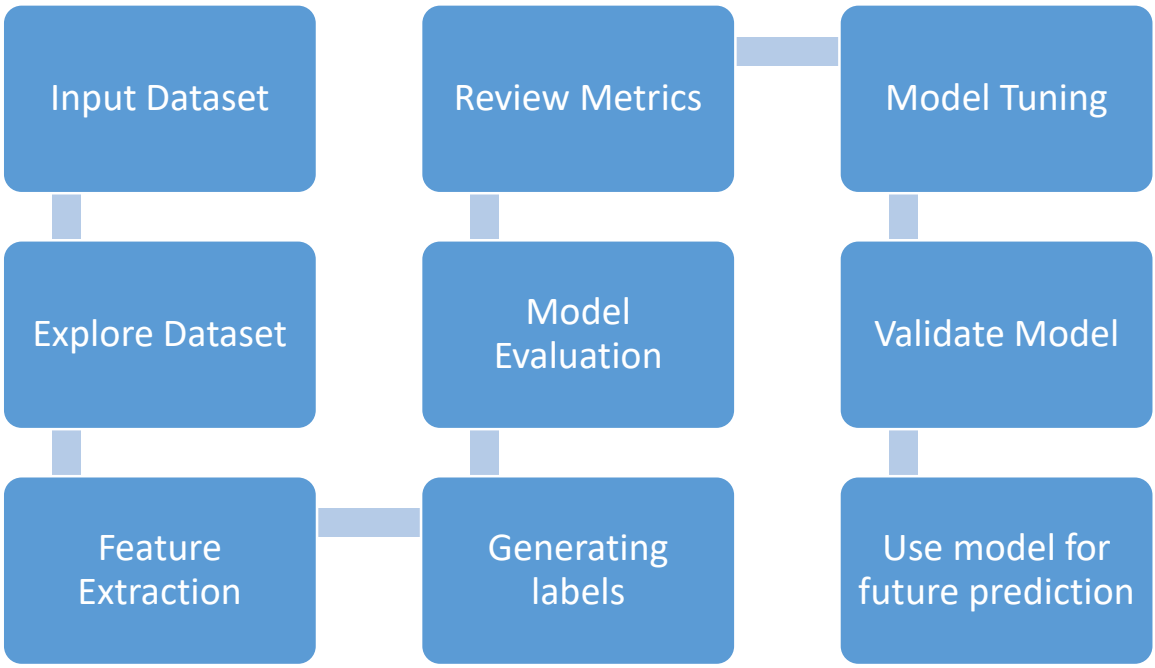
(TP) **130,154**	(FP) **238,793**
(FN) **164,973**	(TN) **2,466,080**

Reflection

After doing initial data exploration I was stuck how data provided can be used to make prediction, then came across feature extraction and construction, which is a must for this project to provide right input data for model and create training-labels out of those extracted data.

Once the data was ready, the next challenge was to pick right algorithm that best suit the problem – With prior experience and knowledge I choose Deep Neural Network with different hyperparameters until we arrive at best score.

Below diagram describes the pipeline employed:



Improvement

We could improve performance of our model by extracting more complex correlated features, our model can perform better if the data we are using to train with will impact their model in positive way and further using algorithms such as XGBoost which has proven record for out-performing other algorithms we can try to improve prediction.