# ASDP: Analyze, Supervise, Detect and Predict

27.12.2021

Nishant Wadhwani
Machine Learning Engineer

## Overview

Finance industry is one of the leading sectors in the modern age to see a steep rise in the usage of Machine learning. May it be managing assets, fraud management, evaluating levels of risk, automated approval of loans, calculating credit scores, automating trading activities, or providing financial advisory services to investors, Machine Learning is being applied almost everywhere possible which offer streamlined customer experience.

Machine Learning algorithms extract meaningful insights from raw sets of data and provide accurate results. This information is then used to solve complex and data-rich problems that are critical to the banking & finance sector. Further, machine learning algorithms are equipped to learn from data, processes, and techniques used to find different insights.

Why Use Machine Learning in Finance?

Here are some of the reasons why banking and financial services firms should consider using Machine Learning –

-->Enhanced revenues owing to better productivity and improved user experience

-->Low operational costs due to process automation

-->Reinforced security and better compliance

With all these applications, technology has come to play an integral role in many phases of the financial ecosystem. The success of a machine learning project depends more on building efficient infrastructure, collecting suitable datasets, and finally applying the right algorithms.

## Goals

We as Data scientists train machine learning models with existing datasets and then apply well-trained models to real-life situations. In general, the larger the dataset fed, the more accurate are the results. Coincidentally, enormous datasets are very common in the financial services industry. There are petabytes of data on transactions, customers, bills, money transfers, and so on. That is a perfect fit for applying machine learning algorithms.

As the technology evolves and the best algorithms are open-sourced, it’s hard to imagine the future of financial services without machine learning.

Let us talk of FinTech Club which is a highly selective, quality-over-quantity organization. Unlike traditional members clubs with a focus on wealth and status, we aim to assemble a community of members that share a common set of values; a passion for knowledge, the desire to collaborate, and the ambition to leave a lasting impact on the industry.

So, as a group of Data Scientists and ML Engineers in one of the FinTech Clubs, we have planned to offer these solutions to huge financial organizations to solve some generic use-cases. They include:

1. Fraud Detection using Self-Organizing Map
2. Customer Retention Program using Churn-Modeling Classification
3. Loan Approval Prediction using Classification
4. Stock Price Prediction using Recurrent Neural Networks(LSTM)

## Credit Card Fraud Detection using Self-Organizing Map:

### Specifications

One of the massive problems for financial institutions is detecting fraud and one of the foremost reasons to leverage machine learning in this case. According to a Bloomberg report, fraud losses incurred by banks and merchants on all credit, debit, and prepaid general purpose and private label payment cards issued globally accounted for £16.74 billion ($21.84 billion) in 2015.

Machine learning is ideally suited to detecting such fraud mechanisms. This is because ML systems can scan through vast data sets, detect unusual activities, (anomalies), and flag them almost instantly. Machine Learning algorithms are excellent at detecting transactional frauds by analyzing millions of data points that are invisible to the human eye. Furthermore, ML also reduces the number of false rejections and helps improve the precision percentage of real-time approvals. This reduces errors and time spent on correcting these error flags. These models are generally built on the client's behavior on the internet and transaction history.

Apart from the high accuracy feature, ML-powered technology is best suited because of its ability to identify suspicious account behavior and prevent fraud in real-time instead of detecting them after the crime has already been committed. This provides a lot of scope for time improvement.

Of the many,  credit card fraud detection is one of the most successful applications of ML. Financial institutions are generally equipped with monitoring systems that are trained on historical payments data. Algorithm training, validation, and backtesting are based on vast datasets of credit card transaction data.

We will implement an unsupervised deep learning model, which is the self-organizing map to solve fraud detection. We are given a data set that contains information on customers from this bank applying for an advanced credit card. So basically, these pieces of information are the data that customers had to provide when filling the application form. And our mission is to detect potential fraud within these applications. So, by the end of the mission, we have to give an explicit list of the customers who potentially cheated.

So our goal is very explicit, we have to return something, however, to return this something that is the list of potential fraudulent customers, we will not make a supervised deep learning model and try to predict if each customer potentially cheated, yes or no with a dependent variable that has binary values. We are going to do is unsupervised deep learning, which means that we will identify some patterns in high dimensional data sets full of nonlinear relationships. And one of these patterns will be the potential fraud i.e. the customers who potentially cheated.

Firstly this data set is taken from the UCI Machine Learning Repository, it is called the Statlog Australian Credit Approval Data Set.

So, let's see the basic information about this data set. This file concerns credit card applications, all attribute names and values have been changed to meaningless symbols to protect the confidentiality of the data. Okay, so that makes this problem even more complex and difficult to solve for humans, indeed when we see the data set, we feel incapable of detecting any fraud.

So we clearly need a deep learning model to find the frauds. The independent variables are some categorical and continuous independent variables. And inside all these variables are hidden SOM frauds that we have to detect. This model is going to do some kind of customer segmentation to identify segments of customers and one of the segments will contain the customers that potentially cheated.

So, all these customers are the inputs of our neural network. And then what happens is that these input points are going to be mapped to a new output space. And between the input space and the output space, we have this neural network composed of neurons, each neuron being initialized as a vector of weights that is the same size as the vector of customer that is a vector of 15 elements, because we have the customer ID plus 14 attributes.

And so for each observation point that is for each customer, the output of this customer will be the neuron that is the closest to the customer.

This neuron is called the winning node, for each customer, the winning node is the most similar neuron to the customer. We will use a neighborhood function like the gaussian neighborhood function, to update the weight of the neighbors of the winning node to move them closer to the point. And we do this for all the customers in the input space. We'll repeat this again. And each time we repeat it, the output space decreases and loses dimensions. It reduces its dimension little by little. And then it reaches a point where the neighborhood stops decreasing, where the output space stops decreasing.

And that's the moment where we obtained our self-organizing map in two dimensions with all the winning nodes that were eventually identified. So now we're getting closer to the frauds. We have to think about outliers because fraud is defined by something that is far from the general rules. The rules must be respected when applying for a credit card. So the frauds are the outlying neurons in this two-dimensional self-organizing map, simply because the outline neurons are far from the majority of neurons that follow the rules.

And so now the question is, how can we detect the outline neurons in the self-organizing map?

->Well, for this, we need the MID, the Mean Interneuron Distance. That means that in our self-organizing map for each neuron, we're going to compute the mean of the Euclidean distance between this neuron and the neurons in its neighborhood. And so what neighborhood is that? Well, we have to define it manually. But we define a neighborhood for each neuron. And we compute the mean of the Euclidean distance between this neuron that we picked and all the neurons in the neighborhood that we defined. And by doing that we can detect outliers because outliers will be far from all the neurons in their neighborhood.

And so this is this exact information that we'll get on the self-organizing map at the end of this part that will allow us to detect outliers, and therefore, to detect fraud. And then we'll use an inverse mapping function to identify which customers originally in the input space are associated with this winning node, which is an outlier. We can clearly distinguish the customers who didn't approve their application and the customers who got approval, because then that will be useful, for example, if we want to detect in priority, the fraudulent customers who got their applications approved, that would make more sense.

Algorithm:

STEP 1: We start with a dataset composed of n features independent variables.

STEP 2: We create a grid composed of nodes, each one having a weight vector of n_features elements.

STEP 3: Randomly initialize the values of the weight vectors to small numbers close to 0 (but not 0).

STEP 4: Select one random observation point from the dataset.

STEP 5: Compute the Euclidean distances from this point to the different neurons in the network.

STEP 6: Select the neuron that has the minimum distance to the point. This neuron is called the winning node.

STEP 7: Update the weights of the winning node to move it closer to the point.

STEP 8: Using a Gaussian neighborhood function of mean the winning node, also update the weights of the winning node neighbors to move them closer to the point. The neighborhood radius is the sigma in the Gaussian function.

STEP 9: Repeat Steps 1 to 5 and update the weights after each observation (Reinforcement Learning) or after batch observations (Batch Learning), until the network converges to a point where the neighborhood stops decreasing.

## Milestones

Fraud Customer IDs

15668679

15636521

15567834

15672912

15646190

15631267

15611189

15762799

15716347

15687688

15638610

15649160

15621244

15632789

15720725

15591035

15694677

15772941

15608804

15698522

15715750

15789201

15723884

15735837

15778290

15751137

15704315

15800049

15785367

15801817

15781574

15725002

15707681

15690169

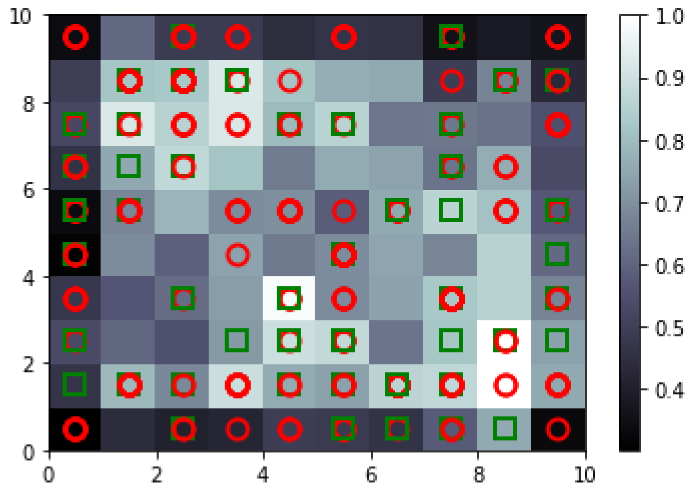
15609823

15594133

15735572

15658504

15569595



SOM which shows fraudulent customers

And now, we did our job, we gave the list of potential cheaters to the bank. Their analysts will investigate this list of potential cheaters and will find out if the customer really cheated somehow.

## Customer Retention Program using Churn-Modeling Classification:

### Specifications

Credit card companies can deploy ML technology to predict the at-risk customers category and specifically retain them. Based on user demographic data and transaction activity we can use past customer data to predict future customer behavior. This would help to design customized offers to retain the customer category.

The application here includes a predictive, binary classification model to find out the customers at risk, which is followed by utilizing a recommender model to identity the best-suited card offers and deals that can help to retain these customers.

The dataset that we will be using contains these columns, it's got row number, customer ID, surname. It's a dataset of a bank containing data of 10000 customers. And what the bank did is they measured some things about these customers, the bank has been seeing unusual churn rates. Churn is when people leave the company and they've seen customers leaving at unusually high rates. So, they want to understand what the problem is and they want to assess and address that problem.

And how did this dataset come to be?

Well, six months ago the bank said alright there's a big problem. This bank has millions of customers, it operates, this fictional bank operates in Europe and three countries France, Spain, and Germany. So what they did is they took this sample of 10 thousand customers that they measured six months ago. Everything they knew about them, their customer ID, their surname, credit score, their geography, their agenda, their age, their tenure, how long they've been with the bank, the balance of the customers then, the number of products they had then, etc.

So the number of products is things like how many products do they have, do they have a savings account, do they have a credit card, do they have a loan, is the customer an active member with a yes/no flag.

An active member can be measured differently by different organizations, it could be whether or not the customer logged onto their online banking in the past month, whether they did a transaction in the past two months, or some other measure like that. And about the estimated salary, the bank doesn't know the salary of the customers but based on the other things they know they could estimate a salary for that customer and they also give you this information.

So six months ago they measured all of these things and said alright so for these 10 thousand randomly selected customers, we're going to just watch them. And six months down the track we're going to check which of these customers left and which of these customers stayed. And that's what this column Exited represents, which will be the dependent variable. It tells you whether or not the person left the bank within those six months.

And our goal is to create a demographic-centric model to tell the bank which of the customers are at the highest risk of leaving. It is such a value-added to any customer-centric organization. So whenever an organization deals with customers, this is a lot of value-added. Any demographic segmentation model can be applied to millions of scenarios. So here for instance, even in a bank, this same scenario could work.

The more customers they have, the more they have money in the bank and the more they make money from the diverse products they offer to their customers. And therefore, they made this data set to understand the reasons somehow why customers leave the bank. And mostly once they managed to build a predictive model that can predict if any new customer leaves the bank. Well, they will deploy this model on new customers. And for all the customers where the model predicts that the customer leaves the bank, they will be prepared and they might do some special offer to the customers so that they will stay in the bank.

So, the bank has asked you as the data scientist to make this predictive model, to first train it on this dataset, to understand the correlations of all these features, and then to deploy this model on future customers.

We will first do exploratory data analysis to get an idea about the data we have, to understand the relationship between different features, and also based upon the plots we will try to anticipate what can be the final answer for the problem so that even the board members in the organization can understand it better. After that, we will do data pre-processing that includes data cleaning, splitting of data, feature scaling, and encoding the categorical data. After that, we will start with the building of machine learning algorithms.

We will use twelve types of classification techniques which include Logistic Regression, KNearestNeighbour Classification, Naive Bayes Classification, Decision Tree Classification, Random Forest Classification, Support Vector Machine Classification, XGBoost Classification, Gradient Boosting Classification, AdaBoost Classification, LightGBM Classification, Catboost Classification, and a Deep Learning Neural Network with two hidden layers. To improve the model performance, we will also apply model selection techniques like K-Fold Cross-Validation and Hyperparameter Tuning using Randomized Search and Grid Search. In the end, we will also compare these models by comparing their accuracies to get the best model.

## Milestones

Results Obtained:-

1. Logistic Regression:-

Accuracy: 80.83 %

Standard Deviation: 0.89 %

```
         precision    recall  f1-score   support
0       0.83       0.96       0.89       1595
1       0.58       0.24       0.34        405
```

2. KNearestNeighbour Classification:-

Accuracy: 82.55 %

Standard Deviation: 0.65 %

```
  precision   recall  f1-score   support
0       0.86      0.93      0.90      1595
1       0.61      0.41      0.49       405
```

3. SVM Classification:-

Accuracy: 85.30 %

Standard Deviation: 1.06 %

```
  precision   recall  f1-score   support
0       0.88      0.96      0.92      1595
1       0.76      0.48      0.59       405
```

4. Naive Bayes Classification:-

Accuracy: 81.69 %

Standard Deviation: 1.35 %

```
  precision   recall  f1-score   support
0       0.86      0.92      0.89      1595
1       0.56      0.41      0.48       405
```

5. Decision Trees Classification:-

Accuracy: 85.08 %

Standard Deviation: 0.80 %

```
    precision   recall  f1-score   support
0       0.88      0.95      0.91      1595
1       0.71      0.51      0.59       405
```

6. Random Forest Classification:-

Accuracy: 85.96 %

Standard Deviation: 0.93 %

```
  precision    recall  f1-score   support
0       0.89      0.96      0.92      1595
1       0.76      0.51      0.61       405
```

7. AdaBoost Classification:-

Accuracy: 85.34 %

Standard Deviation: 1.21 %

```
  precision    recall  f1-score   support
0       0.89      0.94      0.92      1595
1       0.71      0.56      0.62       405
```

8. Gradient Boosting Classification:-

Accuracy: 86.36 %

Standard Deviation: 1.11 %

```
  precision    recall  f1-score   support
0       0.89      0.96      0.92      1595
1       0.76      0.52      0.61       405
```

9. LightGBM Classification:-

Accuracy: 85.80 %

Standard Deviation: 1.04 %

```
  precision    recall  f1-score   support
0       0.89      0.94      0.91      1595
1       0.70      0.54      0.61       405
```

10. XGBoost Classification:-

Accuracy: 86.04 %

Standard Deviation: 0.79 %

```
  precision    recall  f1-score   support
0       0.88      0.96      0.92      1595
1       0.75      0.50      0.60       405
```

11. CatBoost Classification:-

Accuracy: 86.31 %

Standard Deviation: 1.21

```
  precision    recall  f1-score   support
0       0.89      0.95      0.92      1595
1       0.73      0.53      0.61       405
```

12. Deep Learning Neural Network:-

Accuracy: 85.9%

```
  precision    recall  f1-score   support
0       0.89      0.94      0.91      1595
1       0.70      0.54      0.61       405
```

So, as we can see, Gradient Boosting Classifier and Cat Boost Classifier have correctly predicted the result with the highest accuracy i.e. 86.36% and 86.31% as well as good precision, recall value.

Predicting the result of a single observation

Use our ANN model to predict if the customer with the following informations will leave the bank:

Geography: France

Credit Score: 600

Gender: Male

Age: 40 years old

Tenure: 3 years

Balance: $ 60000

Number of Products: 2

Does this customer have a credit card ? Yes

Is this customer an Active Member: Yes

Estimated Salary: $ 50000

So, should we say goodbye to that customer ?

Solution

```
print(ann.predict(sc.transform([[1, 0, 0, 600, 1, 40, 3, 60000, 2, 1, 1, 50000]])) > 0.5)
```

```
[[False]]
```

Therefore, our ANN model predicts that this customer stays in the bank!

Important note 1: Notice that the values of the features were all input in a double pair of square brackets. That's because the "predict" method always expects a 2D array as the format of its inputs. And putting our values into a double pair of square brackets makes the input exactly a 2D array.

Important note 2: Notice also that the "France" country was not input as a string in the last column but as "1, 0, 0" in the first three columns. That's because of course the predict method expects the one-hot-encoded values of the state, and as we see in the first row of the matrix of features X, "France" was encoded as "1, 0, 0". And be careful to include these values in the first three columns, because the dummy variables are always created in the first columns.

## Loan Approval Prediction using Classification:

### Specifications

For this task, we will be exploring publicly available data from LendingClub.com. Lending Club connects people who need money (borrowers) with people who have money (investors). Hopefully, as an investor, you would want to invest in people who showed a profile of having a high probability of paying you back. We will try to create a model that will help predict this. The lending club had a very interesting year in 2016, so let's check out some of their data and keep the context in mind. This data is from before they even went public. We will use lending data from 2007-2010 and be trying to classify and predict whether or not the borrower paid back their loan in full.

Here are what the columns represent:

credit. policy: 1 if the customer meets the credit underwriting criteria of LendingClub.com, and 0 otherwise.

purpose: The purpose of the loan (takes values "credit_card", "debt_consolidation", "educational", "major_purchase", "small_business", and "all_other").

int.rate: The interest rate of the loan, as a proportion (a rate of 11% would be stored as 0.11). Borrowers judged by LendingClub.com to be riskier are assigned higher interest rates.

installment: The monthly installments owed by the borrower if the loan is funded.

log.annual.inc: The natural log of the self-reported annual income of the borrower.

dti: The debt-to-income ratio of the borrower (amount of debt divided by annual income).

fico: The FICO credit score of the borrower.

days.with.cr.line: The number of days the borrower has had a credit line.

revol.bal: The borrower's revolving balance (amount unpaid at the end of the credit card billing cycle).

revol.util: The borrower's revolving line utilization rate (the amount of the credit line used relative to total credit available).

inq.last.6mths: The borrower's number of inquiries by creditors in the last 6 months.

delinq.2yrs: The number of times the borrower had been 30+ days past due on a payment in the past 2 years.

pub.rec: The borrower's number of derogatory public records (bankruptcy filings, tax liens, or judgments).

We want to predict whether or not the loan was fully paid back or not. Then what we are going to do is just do some exploratory data analysis creating some histograms based on the various features and then we are also going to start setting up the data and will need to deal with categorical features so there will be some data cleaning.

We will first do exploratory data analysis to get an idea about the data we have, to understand the relationship between different features, and also based upon the plots we will try to anticipate what can be the final answer for the problem so that even the board members in the organization can understand it better. After that, we will do data pre-processing that includes data cleaning, splitting of data, feature scaling, and encoding the categorical data. After that, we will start with the building of machine learning algorithms.

We will use twelve types of classification techniques which include Logistic Regression, KNearestNeighbour Classification, Naive Bayes Classification, Decision Tree Classification, Random Forest Classification, Support Vector Machine Classification, XGBoost Classification, Gradient Boosting Classification, AdaBoost Classification, LightGBM Classification, Catboost Classification, and a Deep Learning Neural Network with two hidden layers. To improve the model performance, we will also apply model selection techniques like K-Fold Cross-Validation and Hyperparameter Tuning using Randomized Search and Grid Search. In the end, we will also compare these models by comparing their accuracies to get the best model.

## Milestones

Results Obtained:-

1. Logistic Regression:-

Accuracy: 83.93 %

Standard Deviation: 0.21 %

```
          precision    recall  f1-score   support
0      0.84      0.99      0.91      2014
1      0.35      0.02      0.03       381
```

2. KNearestNeighbour Classification:-

Accuracy: 83.98 %

Standard Deviation: 0.13 %

```
  precision    recall  f1-score   support
0      0.84      1.00      0.91      2014
1      0.40      0.01      0.01       381
```

3. SVM Classification:-

Accuracy: 83.96 %

```
Standard Deviation: 0.05 %
  precision    recall  f1-score   support
0       0.84      1.00      0.91      2014
1       0.00      0.00      0.00       381
```

4. Naive Bayes Classification:-

```
Accuracy: 77.59 %
Standard Deviation: 1.52 %
  precision    recall  f1-score   support
0       0.87      0.87      0.87      2014
1       0.30      0.31      0.31       381
```

5. Decision Trees Classification:-

```
Accuracy: 83.99 %
Standard Deviation: 0.23 %
    precision    recall  f1-score   support
0       0.84      1.00      0.91      2014
1       0.43      0.01      0.02       381
```

6. Random Forest Classification:-

```
Accuracy: 83.92 %
Standard Deviation: 0.36 %
  precision    recall  f1-score   support
0       0.84      1.00      0.91      2014
1       0.47      0.02      0.04       381
```

7. AdaBoost Classification:-

```
Accuracy: 83.91 %
```

Standard Deviation: 0.36 %

```
  precision    recall  f1-score   support
0       0.85      0.99      0.91      2014
1       0.38      0.05      0.08       381
```

8. Gradient Boosting Classification:-

Accuracy: 83.80 %

Standard Deviation: 0.25 %

```
  precision    recall  f1-score   support
0       0.84      0.99      0.91      2014
1       0.50      0.03      0.06       381
```

9. LightGBM Classification:-

Accuracy: 83.60 %

Standard Deviation: 0.53 %

```
  precision    recall  f1-score   support
0       0.85      0.99      0.91      2014
1       0.50      0.06      0.11       381
```

10. XGBoost Classification:-

Accuracy: 82.74 %

Standard Deviation: 0.48 %

```
  precision    recall  f1-score   support
0       0.85      0.97      0.91      2014
1       0.40      0.12      0.18       381
```

11. CatBoost Classification:-

Accuracy: 83.66 %

Standard Deviation: 0.49 %

```
  precision    recall  f1-score   support
0       0.84      0.99      0.91      2014
1       0.43      0.03      0.06       381
```

12. Deep Learning Neural Network:-

Accuracy: 83.173%

```
  precision    recall  f1-score   support
0       0.84      0.98      0.91      2014
1       0.32      0.05      0.09       381
```

So, as we can see, the Decision Tree Classifier and KNN classifier have correctly predicted the result with the highest accuracy i.e. 83.99% and 83.98% as well as good precision, recall value. Now, we can correctly determine whether to approve the loan for an individual or not since we have some features, using these variables we can predict the amount will be fully paid or not. If the amount is paid fully, we can approve the loan for the individual, otherwise we will look at some more parameters to make the decision. Clearly here our machine learning model is keeping our efforts nominal and giving us the optimum results.

## Stock Price Prediction using Recurrent Neural Networks(LSTM):

### Specifications

Machine learning can work wonders in making better trading decisions and understanding the choices. A mathematical model monitors the news, studies the market in real-time and identifies trends and factors that cause the stock prices to fluctuate. We can then derive the information from it to sell, hold, or buy stocks according to its predictions. Machine learning algorithms can analyze thousands of data sources simultaneously, something that is not physically possible by humans.

Machine learning algorithms help human traders get an advantage over the market average. And, given the vast volumes of trading operations, even the smallest advantage results in significant profits.

Machine learning stands out for its feature to predict the future using data from the past. The system analyzes a large set of data and comes up with answers to various future-related questions. This gives machine learning the ability to have market insights that allow the fund managers to identify specific market changes. Henceforth, divergence in the market can be detected much earlier as compared to the traditional investment models.

Well-known financial institutions like JPMorgan, Bank of America, and Morgan Stanley are heavily investing in machine learning technologies to develop automated investment advisors. ML-based solutions and models allow trading companies to make better trading decisions by closely monitoring the trade results and news in real-time to detect patterns that can enable stock prices to go up or down.

We are going to tackle a very challenging problem in this part because we're gonna have to predict the stock price of nothing else than Google. So it will be pretty challenging, especially if we have heard some notions in financial engineering since indeed there is the Brownian Motion that states that the future variations of the stock price are independent of the past. So it's actually impossible to predict exactly the future stock price otherwise we would all become billionaires but it's possible to predict some trends.

So, we are going to predict the upward and downward trends that exist in the Google stock price. And to do so, the model that we will implement will be an LSTM.  And with that goal in mind, some research articles suggest that LSTM is the most powerful model that can do this. It performs better than the traditional RMI model. We are going to implement a simple LSTM, it's gonna be super robust with some high-dimensionality, several layers, it will be a stacked LSTM, then we will add some dropout regularization to avoid overfilling and we will use the most powerful optimizer that we have in the Keras library.

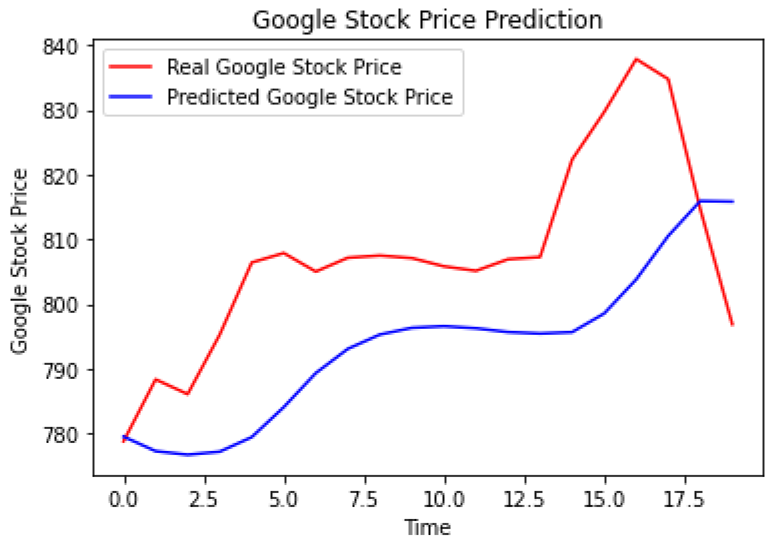
So, we are going to train our LSTM model on five years of the Google stock price, and this is from the beginning of 2012 to the end of 2016 and then, based on the training, based on the correlations identified or captured by the LSTM of the Google stock price, we will try to predict the first month of 2017. And again, we are not going to try to predict exactly the stock price, we are going to predict the trend, the upward or downward trend of the Google stock price.

We are going to train our RNN on only the training sets. The RNN will have no idea of what's going on in a test set. It will have no acquaintance with the test set during its training. It's like the test set doesn't exist for the RNN. But then, once the training is done, we will introduce the test set to the RNN, so that it can make some predictions of the future stock price in January 2017. First, we are going to get the real Google stock price of 2017. In the second step, we are going to get the predicted Google stock price of 2017. And then the third step, the final step, we will visualize the results.

We are going to use a regressor for predicting the Google stock prices of January 2017. Now, the first thing to understand is that we trained our model to be able to predict the stock price at time T plus one, based on the 60 previous stock prices. Therefore, to predict each stock price of each financial day of January 2017, we will need the 60 previous stock prices of the 60 previous financial days, before the actual day.

Now, second key point is that to get at each day of January 2017, the 60 previous stock prices of the 60 previous days, we will need both the training set and the test set, because we will have some of the 60 days that will be from the training set, because they will be from December 2016, and we will also have some stock prices of the test set, because some of them will come from January 2017. Therefore, the first thing we need to do now, is some concatenation of the training set and the test set, to be able to get these 60 produced inputs, for each day of January 2017.

## Milestones



So, first of all, we have the real Google stock price in red and our predicted Google stock price in blue. And we get this comparison of the real and the predicted Google stock prices for the whole month of January 2017. Well, first of all, in some parts of the predictions, we see our predictions lagging behind the actual values. Here we see a big spike, like a stock time singularity and our predictions did not follow that, but that is completely normal. Our model just lags because it cannot react to fast, nonlinear changes. This spike here, this stock time irregularity is indeed a fast, nonlinear change to which our model cannot react properly. But that's fine because, indeed, according to the Brownian Motion Mathematical Concept in financial engineering, the future variations of the stock price are independent of the past. And therefore, this future variation that we see here around the spike, well, is a variation that is indeed totally independent from the previous stock prices.

The good news is that our recurrent neural network model reacts pretty well to these smooth changes. Therefore, the conclusion is that in the parts of the predictions containing some spikes, our predictions lag behind the actual values because our model cannot react to fast, nonlinear changes. And, on the other hand, for the parts of the predictions containing smooth changes, our model reacts pretty well and manages to follow the upward and downward trends. It manages to follow the upward trend here, the stable trend, and again, the upward trend here. Then, there is a downward trend here in the last financial days of January, and it started to capture it.

## Conclusion

Recently, all industries have seen a rapid acceleration in the application of technologies such as AI and ML due to improved software and hardware. The finance sector, specifically, has seen a steep rise in the use cases of machine learning applications to advance better outcomes for both consumers and businesses. From fraud detection, loan approvals, customer retention to stock price prediction, Machine learning is a huge part of banks, trading, fintech firms, financial institutions and so on.

Going forward, the applications of Machine Learning in finance industry are definitely to get creative, to automate time-consuming, mundane processes, and offer a far more streamlined and personalized customer experience.