Stock price direction prediction In this model card project we are using Logistic Regression to predict the direction of price of stock based on the historical data. Below are the mathematical description of Logistic Regression.

Training Dataset

- Alphabet stock(GOOG) pricing data starting from 1st January 2017 to 1st January 2018, training data split: 50% shuffled randomly

Test Dataset

- Alphabet stock(GOOG) pricing data starting from 1st January 2017 to 1st January 2018, test data split: 50% shuffled randomly

Mathematical Framework
Logistic regression is a statistical model which in basic form uses a logistic function to model a binary dependent variable, although there are many more complex extensions.
Mathematically, in a binary logistic model there is a dependent variable with two possible values, such as success/failure which is represented by an indicator variable, where the two values are labeled "0" and "1".

Logistic Model To understand logistic regression lets consider a logistic model with given parameters, then to see how the coefficients can be estimated from data. In this model we have eight predictors, x_1, x_2, x_3, x_4, x_5, x_6 x_7 and x_8 which represents **Price, Return, Lag_1, Lag_2, Lag_3, Lag_4, Momentum and Moving average** respectively, and one binary (Bernoulli) response variable Y, which we denote $p = P(Y = 1)$.

Some of the above predictors can be given as:

$$\begin{aligned} Momentum &= Price_{n+1} - Price_n \\ Return &= logPrice_n - logPrice_{n-1} \\ MovingAverage &= 20 \text{ days rolling average} \end{aligned}$$

Where n is the days of the trade.

We assume a linear relationship between the predictor variables and logit of the event that $Y = 1$. This linear relationship can be written in the following mathematical form, where l is the logit, b is the base of the logarithm, and β_i are parameters of the model:

$$l = log_b \frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6 + \beta_7 x_7 + \beta_8 x_8$$

We can recover the odds by exponentiation the log-odds:

$$\frac{p}{1-p} = b^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6 + \beta_7 x_7 + \beta_8 x_8}$$

By simple algebraic manipulation (and dividing numerator and denominator by $b^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6 + \beta_7 x_7 + \beta_8 x_8}$), the probability that $Y = 1$ is

$$
\begin{aligned}
p &= \frac{b^{\beta_0+\beta_1 x_1+\beta_2 x_2+\beta_3 x_3+\beta_4 x_4+\beta_5 x_5+\beta_6 x_6+\beta_7 x_7+\beta_8 x_8}}{b^{\beta_0+\beta_1 x_1+\beta_2 x_2+\beta_3 x_3+\beta_4 x_4+\beta_5 x_5+\beta_6 x_6+\beta_7 x_7+\beta_8 x_8}+1} \\
&= \frac{1}{1+b^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\beta_3 x_3+\beta_4 x_4+\beta_5 x_5+\beta_6 x_6+\beta_7 x_7+\beta_8 x_8)}} \\
&= S_b(\beta_0+\beta_1 x_1+\beta_2 x_2+\beta_3 x_3+\beta_4 x_4+\beta_5 x_5+\beta_6 x_6+\beta_7 x_7+\beta_8 x_8)
\end{aligned}
$$

Where S_b is the Sigmoid function with base b. The above formula shows that once β_i are fixed, we can easily compute either the logit that $Y = 1$ for a given observation, or the probability that $Y = 1$ for a given observation. The main use-case of a logistic model is to be given an observation $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$, and estimate the probability p that $Y = 1$.

Train complexity of Logistic Regression Big O of Logistic Regression is calculated using the below equation:

$$argmax_w \sum_{i=0}^{n} yi(w^t xi) + b$$

Where,

- w is a vector of size d. Performing the operation $yi(w^t xi)$ takes O(d) steps.
- Iterating it over n data points and finding the maximum sum takes n steps.

So, the time complexity during training a Logistic Regression model is $n(O(d)) = O(nd)$.

To train a Logistic Regression model, we need w and b to find a line(in 2D) or plane/hyperplane(in 3D or more dimension) that can separate both the classes point as perfect as possible so that when it encounter with any new point, it can easily classify, from which class point it belongs to. The value of w and b should be such that it maximize the sum $yi(w^t xi) > 0$.

Run-time/test complexity of Logistic Regression The run-time complexity of Logistic Regression is $O(d)$. Run-time complexity is important because at the end of training, we test our model on unseen data and calculate the accuracy of our model. In case of Logistic Regression, after training the model we get w and b. Given any new point, we have to just perform the operation $w^t xi$. If $w^t xi > 0$, the point is positive and if $w^t xi < 0$, the point is negative. As w is a vector of size d, performing the operation $w^t * xi$ takes $O(d)$ steps as discussed earlier. Hence, Logistic regression is very good for low latency applications, i.e, for applications where the dimension of the data is small.

Space complexity of Logistic Regression While training a Logistic Regression model, we store four things in memory: $x, y, wandb$. As b is a constant, so storing b is just step 1, i.e, $O(1)$ operation. x and y are two matrices of order (nxd) and $(nx1)$ respectively. Storing these two matrices takes $O(nd + n)$ steps. Lastly, w is a vector of size d. Storing it in memory takes $O(d)$ steps. So, the space complexity while training is $O(nd + n + d)$.

After training the model what we just need to keep in memory is w. We just need to perform $w^t x1$ to classify the points. Hence, the space complexity during runtime is in the order of d, i,e, $O(d)$.