Chiltepinos Wings Restaurant Sales Predictions with Linear Regression

Kevin Adriel Woolfolk García (A01251809)
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Querétaro

Keywords: Machine learning, Linear Regression, Statistics, Forecasts, Restaurants.

Abstract: Nowadays machine learning can be implemented to develop forecasts for almost any topic that requires them. A clear example of this is the restaurants, which depend on keeping a clear record of their sales, in order to keep adequate management and projections of their future inventories, supplies and make decisions. In this paper we explain the steps taken in order to apply multiple linear regression (MLP), a widely used statistical data analysis technique, to a dataset for a fast-food restaurant. As an outcome, we managed to obtain a good percentage being above the 80% of accuracy.

1 INTRODUCTION

Restaurants over the world require to have a point of sales in order to adequately manage their recipes, sales, inventory, among other things, having to deal and manage an enormous amount of data inside their database. For means of this project, we chose Chiltepinos Wings database, which is a restaurant located in Sonora, Mexico. They have been open since 2013 being one of the most famous restaurants in the state of Sonora. The prediction of sales on restaurants is a really important feature, which allows them to know the amount of supplies they have to buy for the next month, as well as keep a track and forecasts for their future sales.

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of MLR is to model the linear relationship between the explanatory (independent) variables and response (dependent) variable [1].

Applying MLR is a good way to help the restaurant to forecast sales. Through the use of epochs, which are iterations that help us to perfect and approach to the most accurate parameters possibles.

2 DATASET

The dataset that was used to develop this project was obtained from the database of Chiltepinos Wings. The dataset is composed by 2300 instances, where each instance represents one day. This instances at same time have 6 attributes described below:

- Month: Name of month of that day
- Day: Name of the weekday of that day
- Beads: Number of beads of that day
- Customers: Amount of the customers that went to the restaurant that day
- Articles: Amount of articles that were served that day
- Total: The total is the closed sale of the day

Where month, day, bills, customers and articles are the independent variables and total the dependent variable. The query that was used to get all the information needed for the project was:

```
SELECT
DATENAME(month,apertura) as Month,
DATENAME(dw,apertura) as Day,
(SELECT count(*) FROM cheques WHERE
cheques.idturno = turnos.idturno) as Beads,
(SELECT sum(nopersonas) FROM cheques
WHERE cheques.idturno = turnos.idturno) as
Customers,
(SELECT sum(totalarticulos) FROM cheques
WHERE cheques.idturno = turnos.idturno) as
Articles,
(SELECT sum(totalconpropina) FROM
cheques WHERE cheques.idturno =
turnos.idturno) as Total
FROM turnos ORDER BY idturno
```

When we execute our query we would see our dataset that it show as the following picture:

Resultados | Mensajes

	Month	Day	Beads	Customers	Articles	Total
1	Octubre	Lunes	103	192	451.00	24095.60
2	Octubre	Martes	147	265	683.00	36670.00
3	Octubre	Miércoles	111	225	515.00	29349.16
4	Octubre	Jueves	134	249	617.00	33648.58
5	Octubre	Viernes	205	380	1037.00	58423.76
6	Octubre	Sábado	240	867	1124.00	63844.00
7	Octubre	Domingo	192	354	861.00	49604.08
8	Octubre	Lunes	85	161	412.00	21259.92

Figure 1. Query output

2.1 One-Hot Encoding

In order to use linear regression in our model, we need to interpret the attributes “month” and “day” as numerical values. For this mean, we need to apply One-Hot Encoding into this two attributes, which allows us to convert them into binary numerical values.

The steps followed in order to apply this process are described below:

We import our dataset with pandas.

```
sales = pd.read_csv('train.csv')
```

We get the attribute that we want to one-hot and make them into a list.

```
samples = sales.iloc[:, 0].values
samples = samples.tolist()
```

The list has to be converted into a data frame.

```
df = pd.DataFrame(samples)
```

The previous step is used to get one-hot encoding from columns.

```
one_hot = pd.get_dummies(df[0])
```

Convert the 12 columns with the name of the months

```
df = df.drop(0,axis = 1)
```

Join the encoded Data Frame with the 12 columns that were made before

```
df = df.join(one_hot)
```

This is how it looks our attribute "Month" with One-Hot Encoding:

```
      Abril  Agosto  Diciembre  Enero  Febrer
0         0       0          0      0
1         0       0          0      0
2         0       0          0      0
3         0       0          0      0
4         0       0          0      0
...     ...     ...        ...    ...     ..
1947      0       0          0      0
1948      0       0          0      0
1949      0       0          0      0
1950      0       0          0      0
1951      0       0          0      0
```

Figure 2. Outcome of One-Hot Encoding

This shows us how the attribute change from 1 column to 12 columns.

2.2 Scaling

Once we have all our data in numerical values, we need to scale our data so the gradient descent is able to converge. Scaling of data may be useful or necessary under certain circumstances like in this case we use mean scaling to normalize our data.

Using the next formula:

$$X_{new} = (X_{current} - L_{average}) / L_{max}$$

The values stand for:

- Xcurrent = actual value of x
- Laverage = List average
- Lmax= Maximum value of list
- Xnew = x value scaled

We iterate this formula in all of our attributes and instances until we get scaled all our data.

3 Model

As previously mentioned, we used multiple linear regression for this model so first we need to set our main values as:

- Epochs: 1000
- Learning rate: 0.4

We start by training our model with our training dataset with all the attributes we set before. Once we apply one-hot encoding and scaling to our training dataset we are ready to start.

First we have to call our function Gradients Descents (GD) for each epoch so the parameters can be trained each iteration in order to increase our accuracy.

Gradient descent is one of the most popular algorithms to perform optimization and by far the most common way to optimize neural networks[2]. It is an iterative optimization algorithm used to find the minimum value for a function. We calculate the partial derivatives of the cost function with respect to each parameter and store the results in a gradient.

$$\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Figure 3. Cost function.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

Figure 4. Gradient Descent function.

In order to use the Gradient Descent function on our code we pass the following variables:

- Params: Parameters that would be trained each iteration to forecast the sales.
- Samples: This is the dataset of all the independent values.
- Y: This is the dependent value that we want to predict.
- Alfa: Learning rate.

```
def GD(params, samples, y, alfa):
 temp = list(params)
 general_error=0
 for j in range(len(params)):
  acum =0; error_acum=0
  for i in range(len(samples)):
   error = h(params,samples[i])- y[i]
   acum = acum + error*samples[i][j]
   meanS = alfa*(1/len(samples))*acum
   temp[j] = params[j] - meanS
 return temp
```

We simulate Gradient Descent with our cost function, which it is function h:

```
def h(params, sample):
   acum = 0
   for i in range(len(params)):
    acum = acum + params[i]*sample[i]
   return acum
```

We need to follow on our model error for each iteration, to know if its actually improving we calculate our MSE to know the error of each epoch.

The mean squared error tells you how close a regression line is to a set of points. It does this by taking the distances from the points to the regression line (these distances are the “errors”) and squaring them. The squaring is necessary in order to obtain data without any negative signs, as well as giving more weight to larger differences. It’s called the mean squared error as you’re finding the average of a set of errors.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2$$

Figure 5. Mean Squared Error function.

The way we calculate our MSE per iteration is the following:

```
error_acum = 0
mape=0
for i in range(len(samples)):
  hyp = h(params, samples[i])
  error = hyp-y[i]
  mape = mape + abs((error/y[i])*100)
  error_acum = +error**2
mape_error = mape/len(samples)
MEP = error_acum/len(samples)
```

When we finally finish our epochs we show the error of the model so it can show us how it is decreasing between each iteration.

4 Results

Once our model finishes the process and the parameters are trained, we use the mean absolute percentage error (MAPE) to get the percentage error. MAPE is a statistical measure of how accurate a forecast system is, measuring this accuracy as a percentage, and can be calculated as the average absolute percent error for each time period minus actual values divided by actual values [3].

$$\text{MAPE} = \frac{\sum \frac{|A-F|}{A} \times 100}{N}$$

Figure 6. Mean Absolute Percentage Error function.

Once the process of training the model is finished, with our trained dataset we got an accuracy of **86.0363%** with 1900 instances. After the training is finished, the testing of the dataset with 350 instances calculated an **84.6255%** of accuracy.

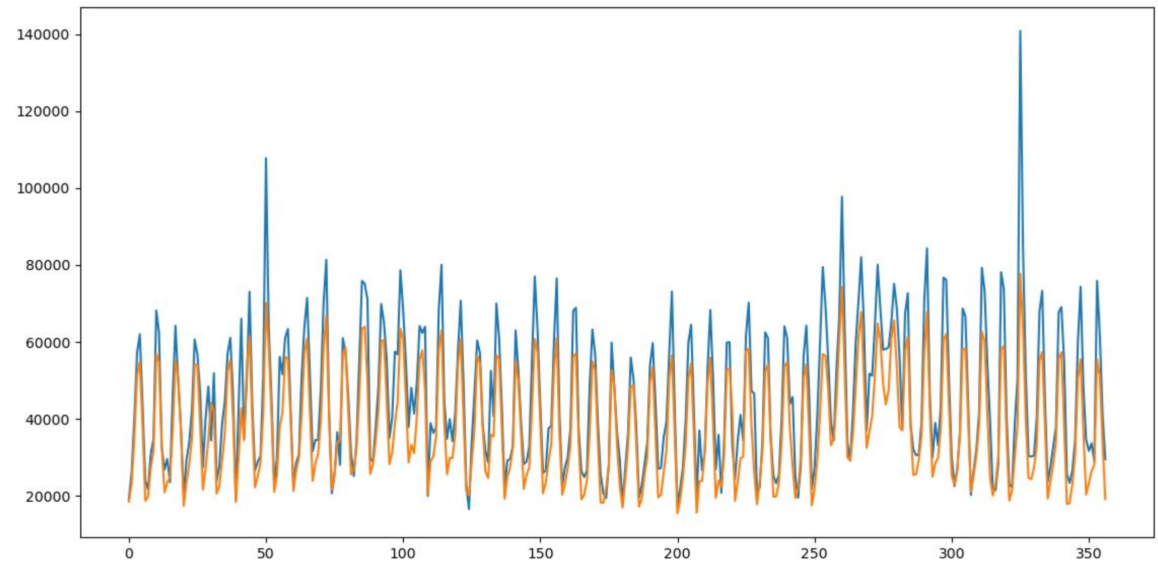

Figure 7. Test Dataset Results

5 Conclusion

According to the accuracy measures obtained from the test dataset results, the forecasts made for the total sales of the restaurant were 84.63% accurate, showing that the multiple linear regression model applied to obtain them, through the use of different techniques was successfully implemented. Through the analysis of the Mean Squared Error and the epochs, we can conclude that the model is overfitting and not decreasing the X values.

It's relevant to clarify that the sales dataset of Chiltepinos Wings is unusual, compared to general sales trends of the average restaurant. This is due to a sudden peak on their sales since day 1 of their opening, contrary to the general trend which is a gradual increase. Therefore, this model is only accurate with Chiltepinos Wings dataset, applying this model to other restaurants won't result on the same accuracy.

The results of this project, can be helpful for the restaurant and the management decision making processes. For the strategic point of view, a projection of the future sales can help on the development of a growth plan,

This forecast model was helpful in order to understand how linear regression works, and how important is the data you give to the model so it can make the prediction. The next step for this is to continue and apply it into an interface for Chiltepinos Wings to apply it for their management.

6 References

1. Investopedia (2019). *Multiple Linear Regression.* Retrieved from: https://www.investopedia.com/terms/m/mlr.asp
2. Pandey, P. (2019). *Understanding the Mathematics behind Gradient Descent.* Retrieved from: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e
3. Statistics How To (2017). *Mean absolute percentage error (MAPE).* Retrieved from: https://www.statisticshowto.com/mean-absolute-percentage-error-mape/
4. Binieli, M. (2018). *Machine learning: an introduction to mean squared error and regression lines.* Retrieved from: https://www.freecodecamp.org/news/machine-learning-mean-squared-error-regression-line-c7dde9a26b93/
5. Fundamentals of Statistics. (2012). *Scaling of Data.* Retrieved from: http://www.statistics4u.com/fundstat_eng/cc_scaling.html
6. Institute of Business Forecasting and Planning. (2020). *MAPE (Mean Absolute Percentage Error).* Retrieved from: https://ibf.org/knowledge/glossary/mape-mean-absolute-percentage-error-174
7. Miller, L. (2018). *Machine Learning week 1: Cost Function, Gradient Descent and Univariate Linear Regression.* Retrieved from: https://medium.com/@lachlanmiller_52885/machine-learning-week-1-cost-function-gradient-descent-and-univariate-linear-regression-8f5fe69815fd