Simple Progression Towards Simple Linear Regression

Introduction :

The goal of the blogpost is to get the beginners started with basics of the linear regression concepts and quickly help them to build their first linear regression model. We will mainly focus on the modeling side of it . The data cleaning and preprocessing parts would be covered in detail in an upcoming post.

Linear Regression are one of the most fundamental and widely used Machine Learning Algorithm. Linear regression is usually among the first few topics which people pick while learning predictive modeling.Linear Regression establishes a relationship between dependent variable (Y) and one or more independent variables (X) using a best fit straight line (also known as regression line).The dependent variable is continuous, independent variable(s) can be continuous or discrete, and nature of regression line is linear.

Linear relationship can either be positive or negative. Positive relationship between two variable basically means that that increase in one variable should also increase the value in other value by some constant value. Negative relationship between two variable means that that increase in one variable should decrease the value in other value by some constant value.

Mathematical Explanation :

A simple linear regression has one independent variable. Mathematically, the line representing a simple linear regression is expressed through a basic equation :

|  |
| --- |
| Y = mX + b + e |

Here ,

m is the slope

X is the predictor variable

b is the intercept/bias term

Y is the predicted target variable

e is the error term

Enough of theory now let’s dive into the implementation of both; a simple linear regression and a multivariate linear regression.

We will use implementation provided by the python machine learning framework known as scikit-learn.

**Problem Statement :**

Predict the price of a car given its compression ratio.

**Data details**

These data sets are originally from the <http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml>.

The label is the identifier of the column you are trying to predict. The identified features are used to predict the label.

**vendor\_id**: The ID of the taxi vendor is a feature.

**rate\_code**: The rate type of the taxi trip is a feature.

**passenger\_count**: The number of passengers on the trip is a feature.

**trip\_time\_in\_secs**: The amount of time the trip took. You won't know how long the trip takes until after it is

**Completed** : You exclude this column from the model.

**trip\_distance**: The distance of the trip is a feature.

**payment\_type**: The payment method (cash or credit card) is a feature.

**fare\_amount**: The total taxi fare paid is the label.

**Tools used** :

Pandas , Numpy , Matplotlib , scikit-learn

**Python Implementation with code :**

**0. Import necessary libraries**

Import the necessary modules from specific libraries.

|  |
| --- |
| from sklearn import linear\_model  import pandas as pd  import matplotlib.pyplot as plt  import numpy as np |

**1. Load the data set**

Use pandas module to read the taxi data from the file system. Check few records of the dataset.

|  |
| --- |
| taxi\_train = "data/taxi-fare-train.csv"  taxi\_test = "data/taxi-fare-test.csv" |

|  |
| --- |
| tax\_train = pd.read\_csv(taxi\_train)  tax\_train.head()  vendor\_id rate\_code passenger\_count trip\_time\_in\_secs trip\_distance payment\_type fare\_amount  0 CMT 1 1 1271 3.8 CRD 17.5  1 CMT 1 1 474 1.5 CRD 8.0  2 CMT 1 1 637 1.4 CRD 8.5  3 CMT 1 1 181 0.6 CSH 4.5  4 CMT 1 1 661 1.1 CRD 8.5 |

**2. Select the predictor feature for Simple Regression , select the target variable**

|  |
| --- |
| X = tax\_train['trip\_distance']  y = tax\_train['fare\_amount'] |

**3. Check 5-num summary of selected predictor feature**

|  |
| --- |
| X.describe()  count 728541.000000 mean 2.741597 std 3.298091 min 0.000000 25% 1.000000 50% 1.700000 75% 3.000000 max 98.700000 Name: trip\_distance, dtype: float64 |

**5. Train test split :**

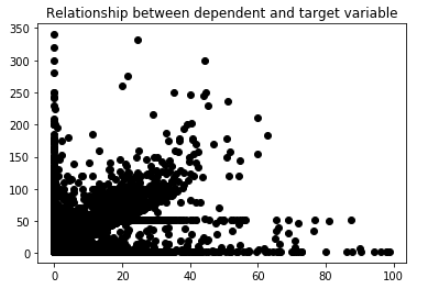
|  |
| --- |
| from sklearn.model\_selection import train\_test\_split  x\_training\_set, x\_test\_set, y\_training\_set, y\_test\_set = train\_test\_split(X,y) |

Dome some reshaping of the variable for visualisation

|  |
| --- |
| x\_training\_set, x\_test\_set, y\_training\_set, y\_test\_set = x\_training\_set.values, x\_test\_set.values, y\_training\_set.values, y\_test\_set.values  x\_training\_set, x\_test\_set, y\_training\_set, y\_test\_set = x\_training\_set.reshape(-1, 1), x\_test\_set.reshape(-1, 1), y\_training\_set.reshape(-1, 1), y\_test\_set.reshape(-1, 1) |

**Do some initial visual inspection between predictor and target variable**

|  |
| --- |
| # So let's plot some of the data  # - this gives some core routines to experiment with different parameters  plt.title('Relationship between dependent and target variable')  plt.scatter(x\_training\_set, y\_training\_set, color='black')  plt.show() |



**6. Training / model fitting**

Fit the model to selected supervised data

|  |
| --- |
| lm = linear\_model.LinearRegression()  lm.fit(x\_training\_set,y\_training\_set) |

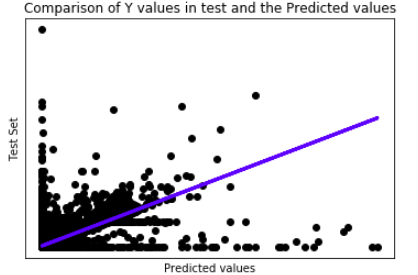
**7. Model parameters study :**

|  |
| --- |
| from sklearn.metrics import mean\_squared\_error, r2\_score  model\_score = lm.score(x\_training\_set,y\_training\_set)  # Have a look at R sq to give an idea of the fit ,  # Explained variance score: 1 is perfect prediction  print('R sq: ',model\_score)  y\_predicted = lm.predict(x\_test\_set)  # The coefficients  print('Coefficients: ', lm.coef\_)  # The mean squared error  print("Mean squared error: %.2f"  % mean\_squared\_error(y\_test\_set, y\_predicted))  # Explained variance score: 1 is perfect prediction  print('Variance score: %.2f' % r2\_score(y\_test\_set, y\_predicted))  ('R sq: ', 0.7729861480277364) ('Coefficients: ', array([[2.55423554]])) Mean squared error: 21.33 Variance score: 0.77 |

**8. Accuracy report with test data :**

Let’s visualise the goodness of the fit with the predictions being visualised by a line

|  |
| --- |
| # So let's run the model against the test data  y\_predicted = lm.predict(x\_test\_set)  plt.title('Comparison of Y values in test and the Predicted values')  plt.xlabel('Test Set')  plt.yabel('Predicted values')  plt.plot(x\_test\_set, y\_predicted, color='blue', linewidth=3)  plt.xticks(())  plt.yticks(())  plt.show() |



**9. Prediction :**

Algo Advantages :

* Extremely simple method
* When relationships between the independent variables and the dependent variable are almost linear, shows optimal results.
* Very easy and intuitive to use and understand
* Even when it doesn’t fit the data exactly, we can use it to find the nature of the relationship between the two variables.

Algo Disadvantages :

* Linear regression is limited to predicting numeric output.
* Very sensitive to the anomalies in the data (or outliers)
* If we have a number of parameters than the number of samples available then the model starts to model the noise rather than the relationship between the variables.
* Regression coefficients are biased by the data imbalance.