Linear Regression in Python

Jose Chavez

July 26, 2023

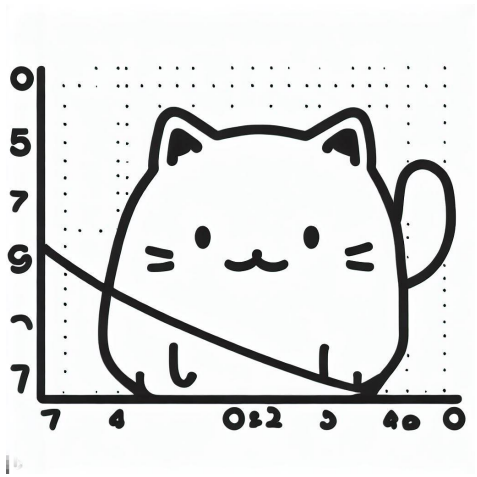

Table of Contents

1. Introduction
2. Data Set: Fake Twitter Data
3. Python and Linear Regression
4. An Example
   - Data Collection

Recap and Introduction

- Linear Regressions: Dependent variable (y) and one or more independent variables ($x_1, x_2, ..., x_n$).
  Fitting a linear equation to the observed data.
  Minimize differences between. real and predicted
  The linear regression model is represented by the equation:

$$y = b_0 + b_1 x_1 + b_2 x_2 + ... + b_n x_n$$

  where:
  - y is the dependent variable (the one we want to predict),
  - b_0 is the intercept (the value of y when all x values are zero),
  - $b_1, b_2, ..., b_n$ are the coefficients (slope) for each independent variable $x_1, x_2, ..., x_n$.
- Simple yet instructive case. . . good old $y = mx + b$

Hypothetical Project

Question:
Are older Twitters user's less likely to use the app throughout the week? What's the relationship between age usage?

Data: Fake Twitter Data

- Hypothetical Data Scientist scenario:
  You are asked to investigate the connection between age and tweet rates.
  You are given an access token to an API.
- Dataset for our demonstration:
  Python scripts you can find at: github were used to generate hundreds of records of consisting of records that include: age, date, tweet (a string);
- Why not the real thing?
  Privacy: Twitter does not realease even that user information.

```
const express = require('express');
const app = express();
const fs = require('fs');
const path = require('path');

app.get('/download', function(req, res){
  const token = req.query.token;

  // Check if the token is valid
  if(token !== 'playtoken'){
    res.status(401).send('Unauthorized: Invalid token');
    return;
  }

  // If the token is valid, serve the CSV file
  const file = path.join(__dirname, 'linear_data.csv');
  res.download(file);
});

app.listen(3000, function(){
  console.log('Server is running on port 3000');
});
```

The Data Science Process

- **Understand the Problem:** Define problem and objective.
  Here: The problem is finding a relationship between the ages and weekly tweet rates.
- **Data Collection:** Gather data.
- **Exploratory Data Analysis (EDA):** Explore the data to get a feel for its strucute.
  Here: We're going to view a scatter plot to get a rough idea whether or not we should expect any success.

The Data Science Processs (2)

- **Feature Selection:** Choosing relevant features (given in our problem).
  Here: This is straighforward in our scenario.
- **Data Splitting:** split to train and to test.
- **Model Building:** Use data to create a model.
- **Model Evaluation:** Assess the model's performance using evaluation metrics like Mean Squared Error (MSE), R-squared, or Root Mean Squared Error (RMSE).

The Data Science Process(3)

- **Model Validation:** Validate the final model on the testing data to ensure it generalizes well to new, unseen data.
- **Interpretation:** Interpret your findings.
- **Deployment:** Deploy the trained model to make predictions on new data or integrate it into a larger application.
- **Monitoring and Maintenance:** Monitor the model's performance, retrain, tune, etc.

Python and Linear Regression

- Idea: Use Python's linear regression library
- Python Libraries:

  Importing and cleaning the data:
  - pandas: data structures to hydrate;
  - python scripts for detecting and removing outliers and null values;

  Exploring the data:
  - matplotlib

  Model training and evaluation:
  - `import sklearn as sk`
  - sk.LinearRegression;
  -

  Visualing the model's results

1. matplotlib

Collect Data from our API

How does text interact with this. Our training data we set aside seems to "align" with the line. . .

ages	rates
54	53.855321
67	59.560101
44	58.634860
30	86.417711
58	46.608163
23	77.859235
36	48.900427

Collect Data from our API (2)

The JavaScript Node.js Application serving the csv file we generated with Python.

ranger (Python) ⌘1 ranger (nvim) ⌘2

```
const fs = require('fs');
const path = require('path');

app.get('/download', function(req, res){
  const token = req.query.token;

  // Check if the token is valid
  if(token !== 'playtoken'){
    res.status(401).send('Unauthorized: Invalid token');
    return;
  }

  // If the token is valid, serve the CSV file
  const file = path.join(__dirname, 'linear_data.csv');
  res.download(file);
});

app.listen(3000, function(){
  console.log('Server is running on port 3000');
});
```

To access: `curl -i http://csv.bluewfjc.online/download?token=playtoken`

Hydrating a DataFrame

We load the csv into a pandas DataFrame.

```
1 import pandas as pd
2 df = pd.read_csv('filename.csv')
```

Examine Visuals of Data

Plotting ages versus rates per week Does it look linear?

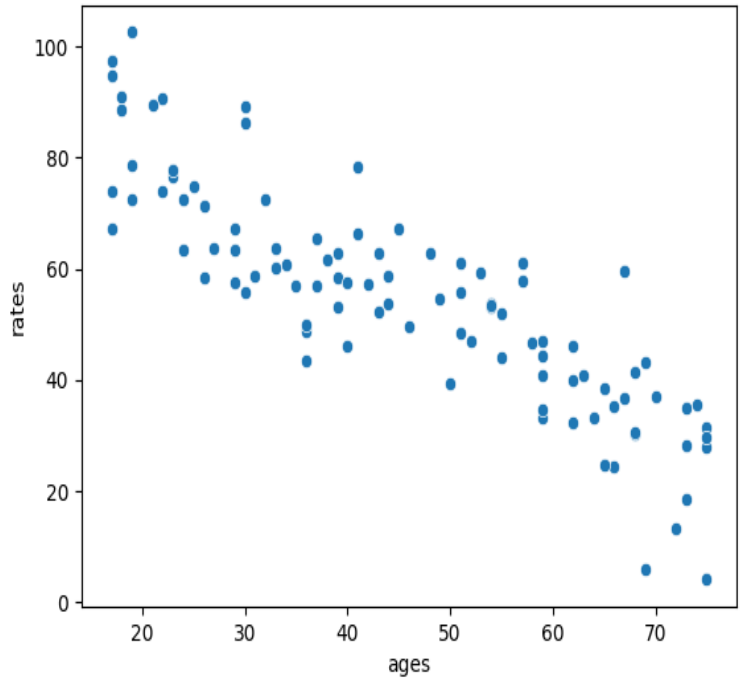

Figure: Scatter Plot of Data Produced with Matplotib

Cleanup

Removes negative 'rates' values

Drops rows with null values

Displays initial cleaned data

Calculates quartiles and IQR

Removes outliers from data

Displays data without outliers

```
  # Drop negative values
df = df[df['rates'] >= 0]
# Drop null values
df = df.dropna()

# Display the first few rows of the
    dataframe
df.head()
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1

# Remove outliers
df_no_outliers = df[~((df < (Q1 - 1.5 *
    IQR)) | (df > (Q3 + 1.5 * IQR))).
    any(axis=1)]

# Display the first few rows of the
    dataframe without outliers
df_no_outliers.head()
```

Figure: Pyt Data Scatter Plot With Line

Clean Up

Code run and differences (post IQR Method)

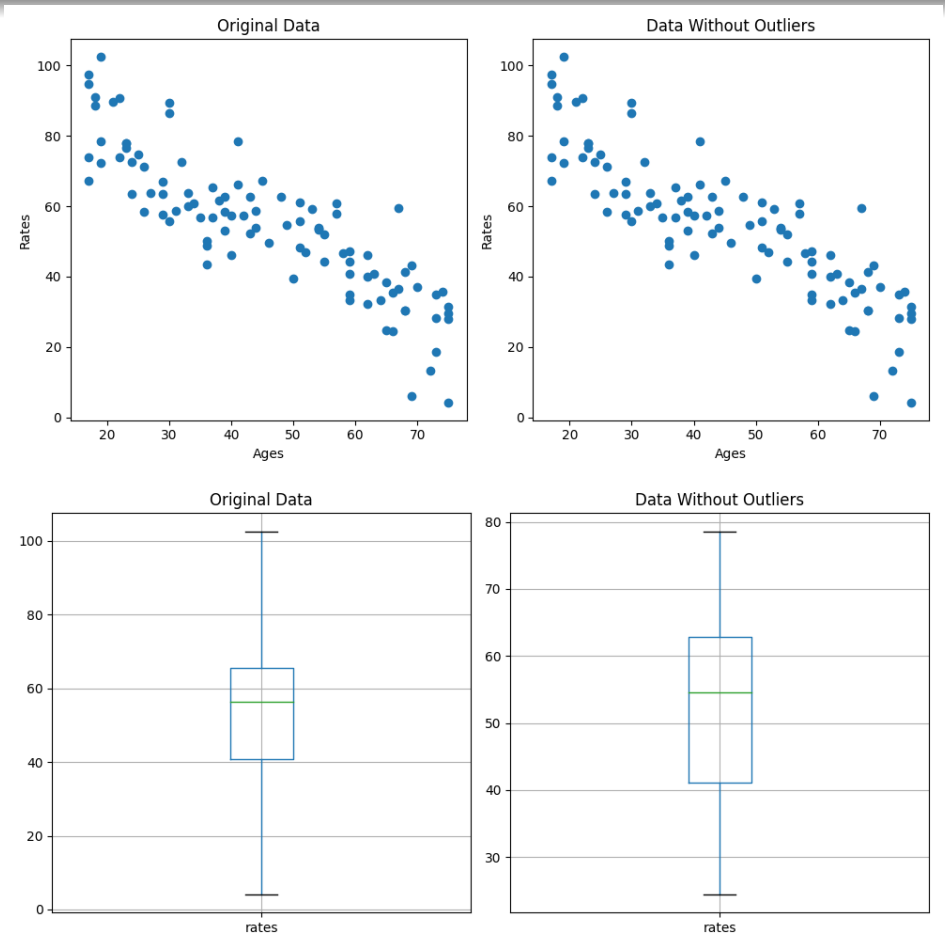

Figure: Box Plot Shows Outliers Have Been Extracted

Library Imports and Splitting the Data

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
  # Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
     random_state=42)

# Create a linear regression object
regr = LinearRegression()

# Train the model using the training sets
regr.fit(X_train, y_train)
```

.

Gathering the Model's Parameters and Testing

```
12 # Make predictions using the testing set
13 y_pred = regr.predict(X_test)
14 # Print the coefficients
15 #print('Coefficients:', regr.coef_)
16 model_cos=regr.coef_
17
18 # Print the mean squared error
19 #print('Mean squared error:', mean_squared_error(y_test, y_pred))
20 mse=mean_squared_error(y_test, y_pred)
21
22 coefficients = regr.coef_[0]
23 intercept = regr.intercept_
24 # Print the coefficient of determination (R^2 score)
25 #print('Coefficient of determination (R^2 score):', r2_score(y_test, y_pred))
26 #r2score=r2_score(y_test,y_pred)
```

The Model We Have Created...

Model Coefficients: Model Intercept (only one):
Mean Squared Error: [-0.74629129]
and MSE (means squared error)
33.912656604887246 -0.7462912894807127
and intercept ...
88.23380257847109
and as a formula:
$y = -0.75x + 88.23$

```
kForLACC$ python3 helpers.py
R2 score: 0.8266009264508798
```

Evaluating the Model Visually

Our training data we set aside seems to "align" with the line...

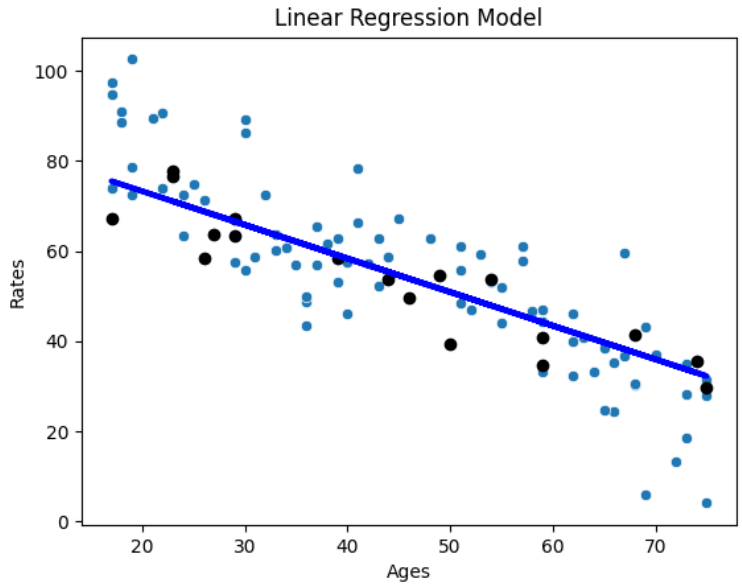

Figure: Training Data Scatter Plot With Line