# Jupyter: Training Linear Models

Submitted by Manish Bafna
Student Id: 19655
Instructor: Dr. Henry Chang
GIT: **Training Linear Model**

# Table of Content

# Introduction

One of the most common statistical methods is linear regression.

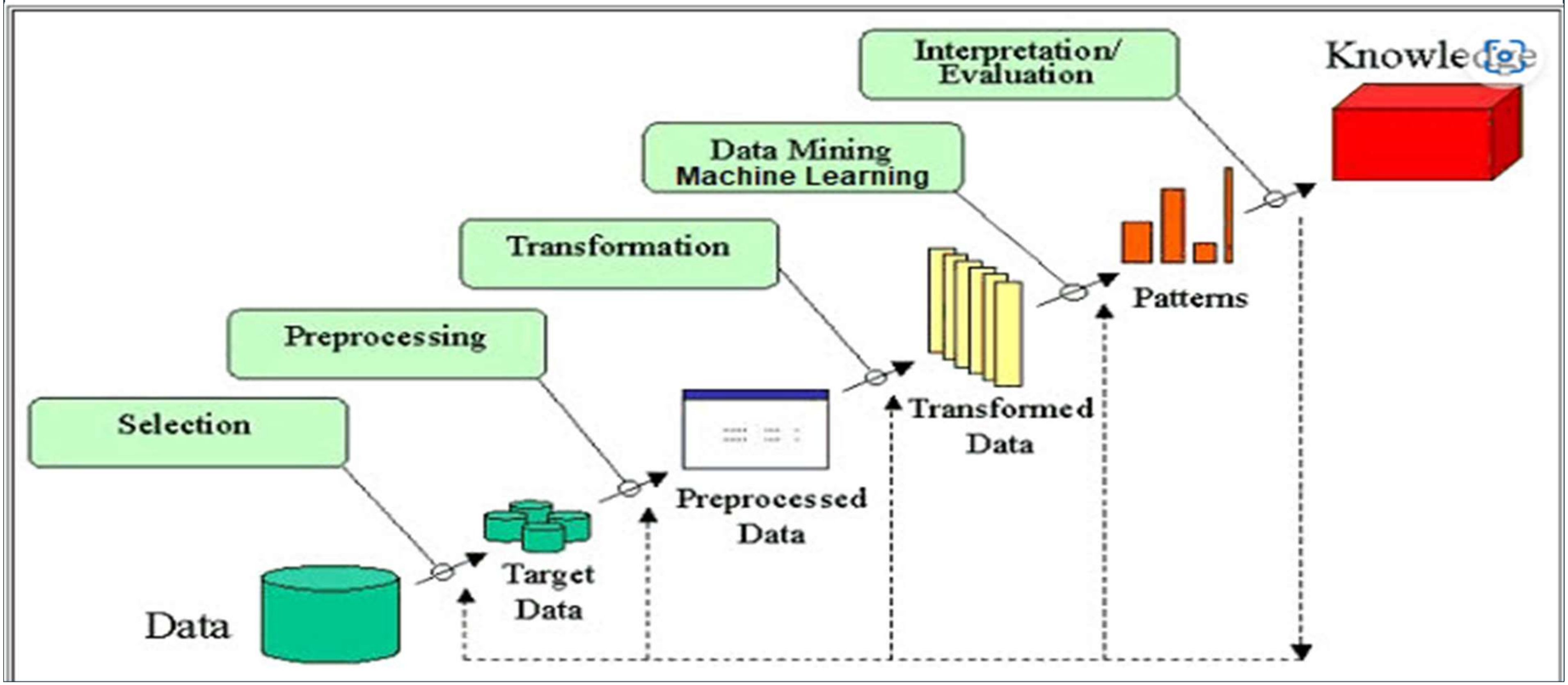
- Regression Analysis helps one understand how the typical value of the dependent variable changes when any one of the independent variables is varied, while the other independent variables are held fixed.
- Linear Regression
  a. Most basic regression form
     i. To express the linear relationship between two variables
        1. Outcome Variable (i.e., dependent varilable)
        2. Predictor (i.e., independent variable)
  b. More sophisticated form
     i. To express the linear relationship between one variable and several other variables
- Linear Regression is an algorithm and a model

# Introduction

- For Linear Regression problems, people typically use a cost function that measures the distance between the linear model's predictions and the training examples; the objective is to minimize this distance.
- This is where the Linear Regression algorithm comes in: you feed it your training examples, and it finds the parameters that make the linear model fit best to your data. This is called *training* the model
- Now the model fits the training data as closely as possible
- You are finally ready to run the model to make predictions
- Linear models are simple and widely used machine learning algorithms that make predictions based on a linear combination of input feature
- The goal of training linear models is to find the optimal values for the models parameters that minimize the prediction error on a given training set
- Common loss functions used in training the linear model include the mean squared error, mean absolute error, hinge loss etc

# Design

Except for "Preprocessing", this exercise invloves all the steps described in the following diagram



# Design

Google Colab

Download the abalone_train.csv file

Modify the python code

Perform Linear Regression Operation from Sample Code

# Implement

Open in Colab

**Chapter 4 – Training Models**

*This notebook contains all the sample code and solutions to the exercises in chapter 4.*

Open in Colab | Open in Kaggle

## Setup

# Implement

```
[890] import numpy as np
      import pandas as pd


      from google.colab import files
      uploaded  = files.upload()
```

```
Choose Files abalone_train.csv
• abalone_train.csv(text/csv) - 145915 bytes, last modified: 2/3/2023 - 100% done
Saving abalone_train.csv to abalone_train (1).csv
```

# Implement

```
import io
abalone = pd.read_csv(
    io.BytesIO(uploaded['abalone_train.csv']),
    names=["Length", "Diameter", "Height", "Whole weight", "Shucked weight",
           "Viscera weight", "Shell weight", "Age"])
```

```
[892] X1 = abalone["Length"]
```

```
[893] X2 = np.array(X1)
```

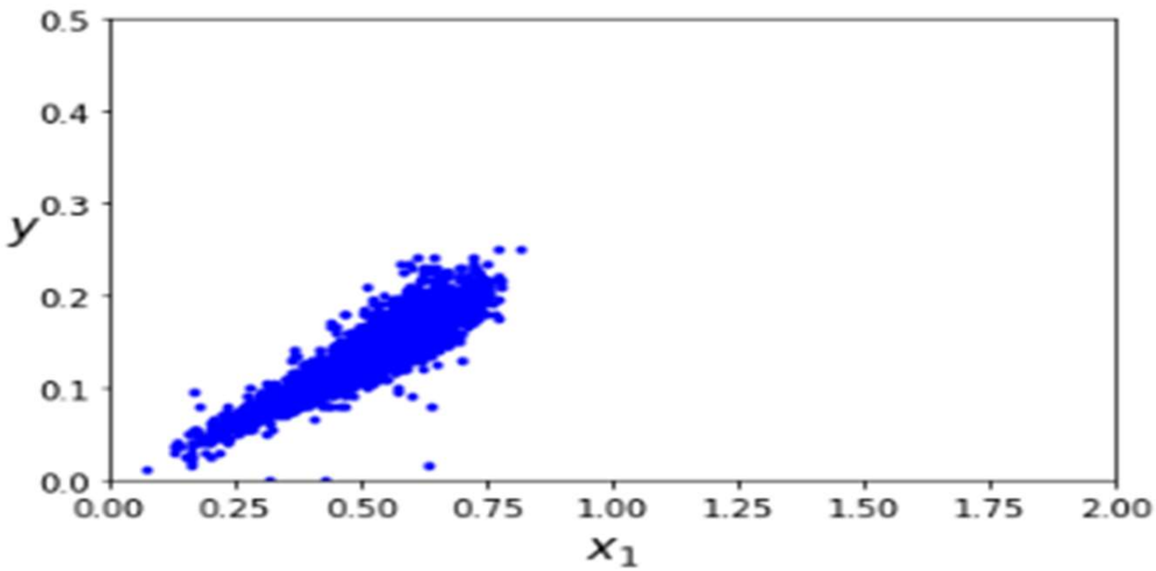
```
[894] X = X2.reshape(-1, 1)
```

```
[895] y1 = abalone["Height"]
      y2 = np.array(y1)
      y = y2.reshape(len(y2), 1)
```

# Test

```
plt.plot(X, y, "b.")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.axis([0, 2, 0, 0.5])
save_fig("generated_data_plot")
plt.show()
```

```
Saving figure generated_data_plot
```

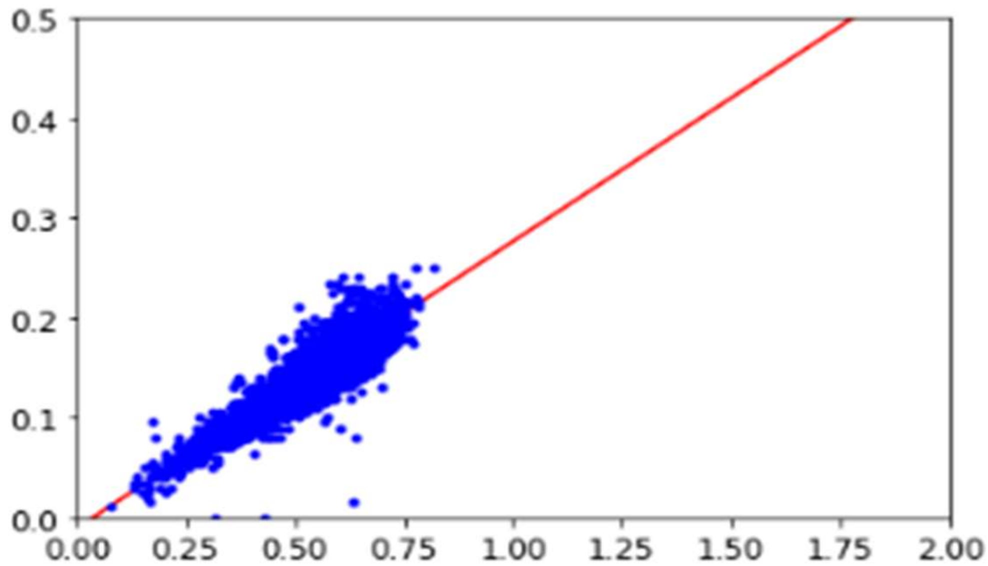


# Test

```
[898] X_b = np.c_[np.ones((len(y), 1)), X]  # add x0 = 1 to each instance
      theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)

[899] theta_best

      array([[-0.0108267 ],
             [ 0.28716253]])

      X_new = np.array([[0], [2]])
      X_new_b = np.c_[np.ones((2, 1)), X_new]  # add x0 = 1 to each instance
      y_predict = X_new_b.dot(theta_best)
      y_predict

      array([[-0.0108267 ],
             [ 0.56349837]])
```

# Test

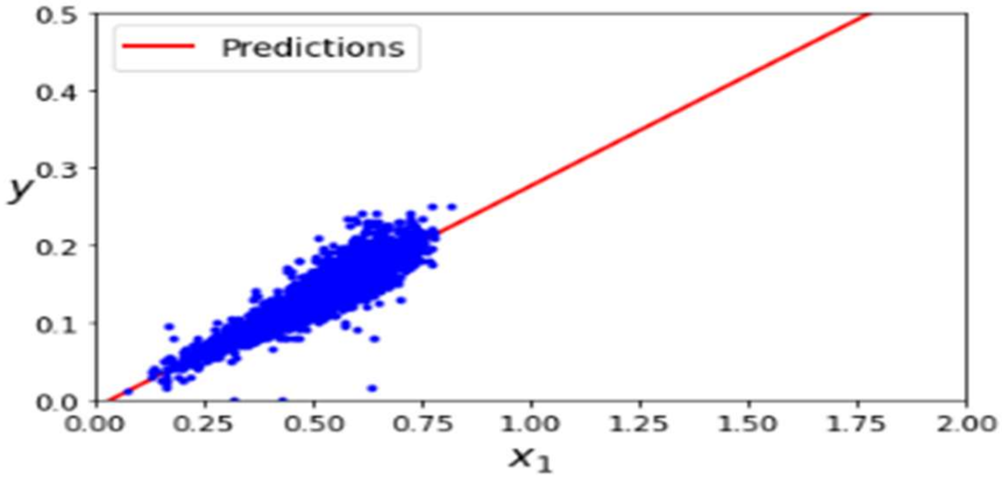
```
plt.plot(X_new, y_predict, "r-")
plt.plot(X, y, "b.")
plt.axis([0, 2, 0, 0.5])
plt.show()
```



# Test

```
plt.plot(X_new, y_predict, "r-", linewidth=2, label="Predictions")
plt.plot(X, y, "b.")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.legend(loc="upper left", fontsize=14)
plt.axis([0, 2, 0, 0.5])
save_fig("linear_model_predictions_plot")
plt.show()
```

```
Saving figure linear_model_predictions_plot
```



# Enhancement Ideas

- Regularization techniques like L1 or L2 regularization, can be applied to normal equation solution to avoid overfitting and improve generalization performance

- Normalizing the input features before performing linear regression can help to improve  the performance and stability of the normal equation solution.

# Conclusion

- It eliminates the need to manually select the learning rate parameter and has a memory usage of O(m), making it suitable of handling large datasets that fit into the computers memory

- This approach provides a simple and effective way for training linear models and can lead to improved performance in various machine learning tasks.

# References


- Exercises for Training Models (sfbu.edu)
- 1. The Machine Learning Landscape - Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition [Book] (oreilly.com)
- Three Basic Algorithms (sfbu.edu)
- The Normal Equation (sfbu.edu)