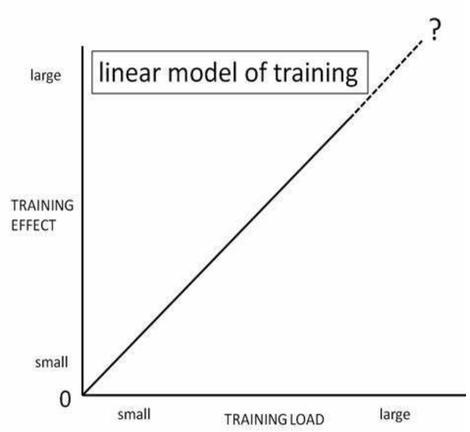
Jupyter: Training Linear Models

CS550 - Machine Learning and Business Intelligence



Submitted by: Divya Pandey(19665)
Instructor: Dr. Henry Chang

Table of Content

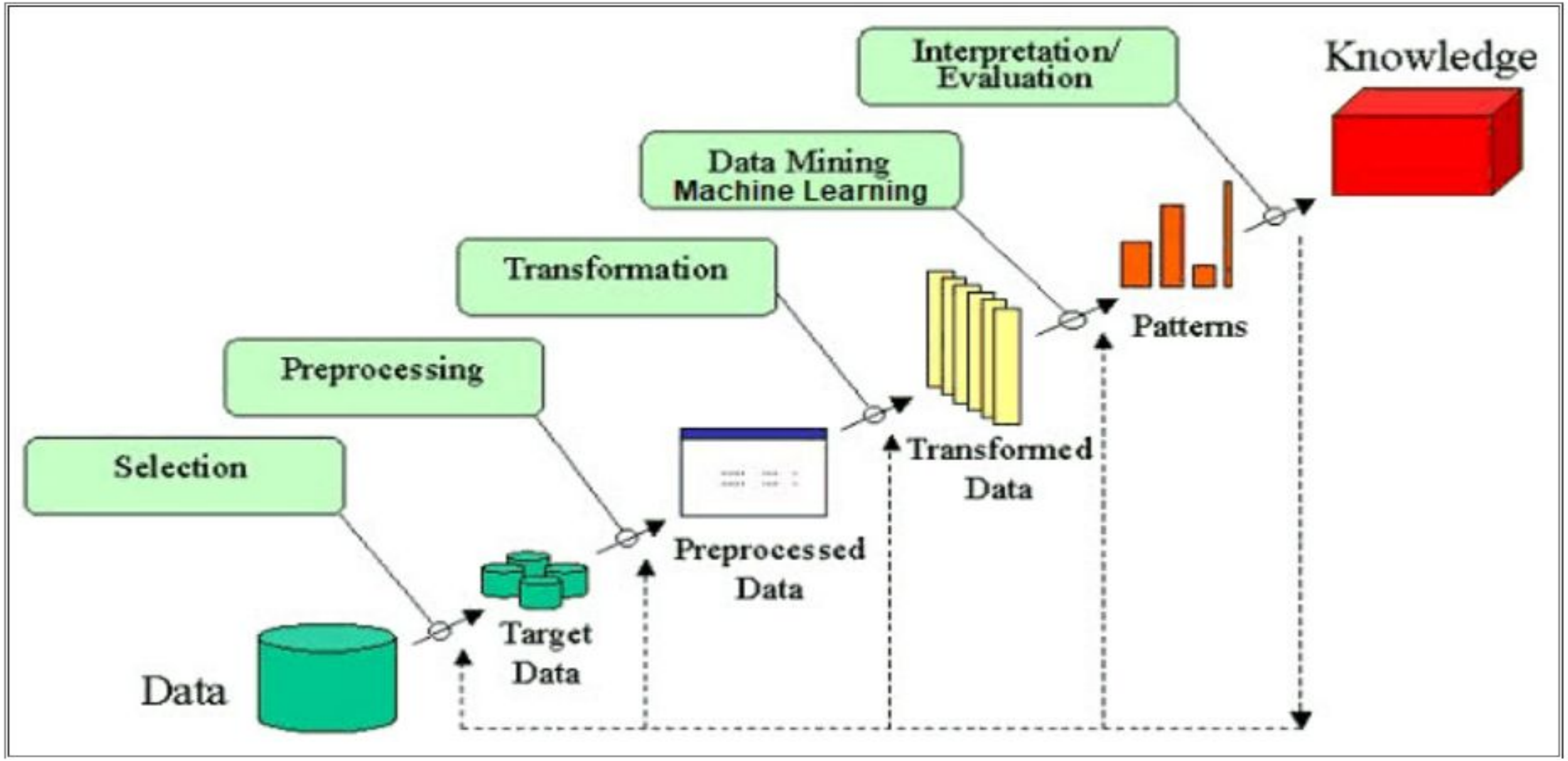
Introduction

★ Linear models are a simple and widely used type of machine learning algorithms that make predictions based on a linear combination of input features.

★ The goal of training linear models is to find the optimal values for the model's parameters that minimize the prediction error on a given training dataset.

★ Common loss functions used in training linear models include mean squared error, mean absolute error, and hinge loss.

Design



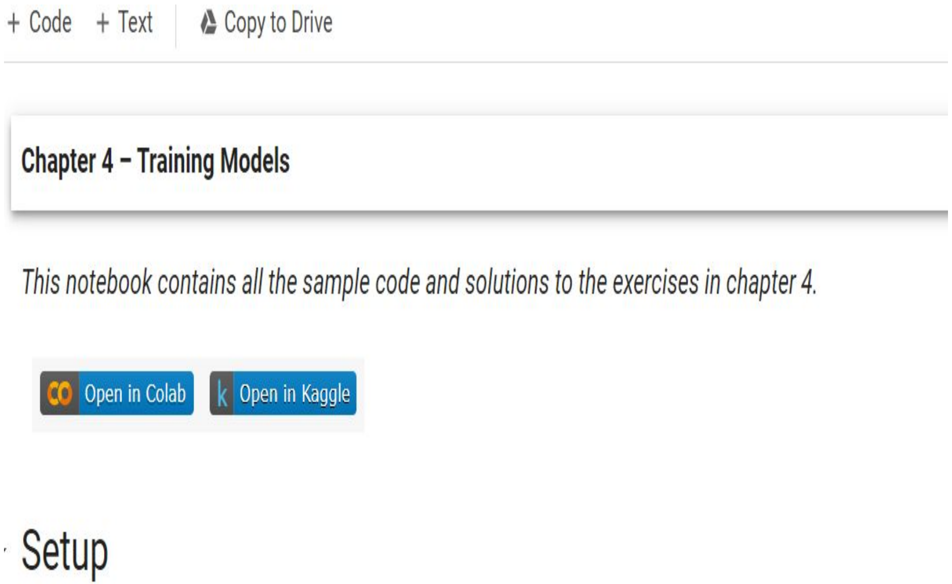
Interpretation/
Evaluation
Knowledge
Data Mining
Machine Learning
Transformation
Patterns
Preprocessing
Transformed
Data
Selection
Preprocessed
Data
Target
Data
Data

Design

- Google Colab
- Download abalone_train.cvs file in your local computer or drive.
- Modified Python Code.
- Perform Linear Regression Operations from Sample Code.

Implementation

Open in Colab



Implementation

Chapter 4 – Training Models

```
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: 2/4/2023 - 100% done

```
Saving abalone_train.csv to abalone_train (2).csv
```

Implementation

```
[19] 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"])
```

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

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

```
X = X2.reshape(-1, 1)
```

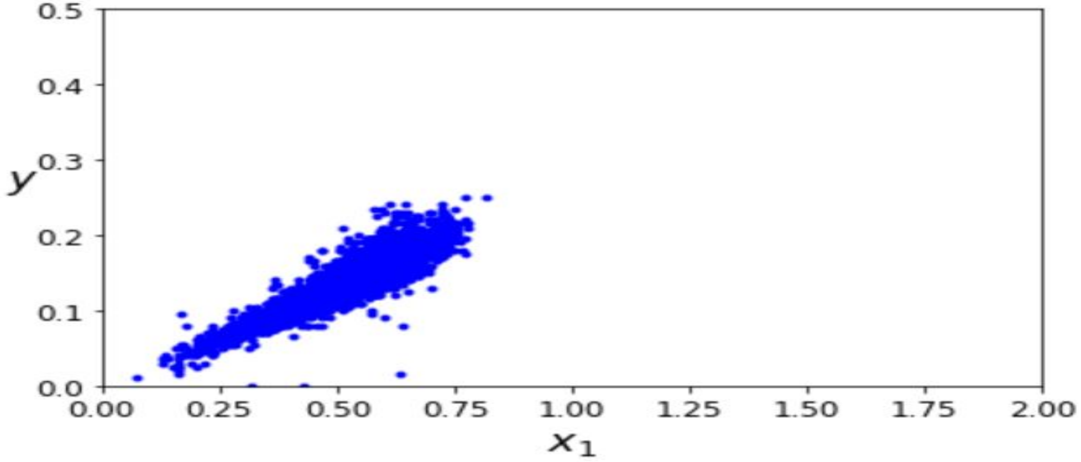
Implementation

```
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

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

```
[37] X_b = np.c_[np.ones((len(y), 1)), X]
     theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)
```

```
theta_best
```

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

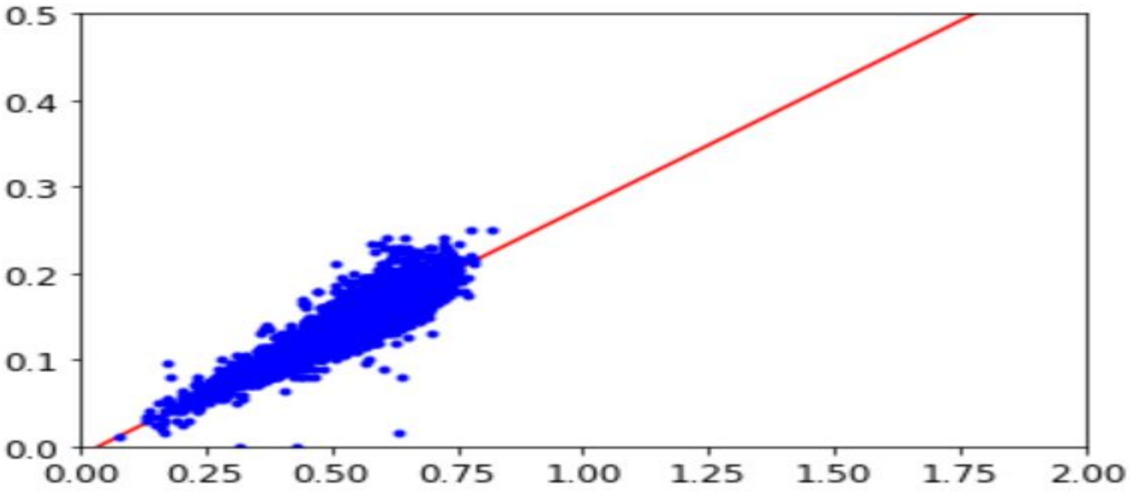
Test

```
[39] 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()
```



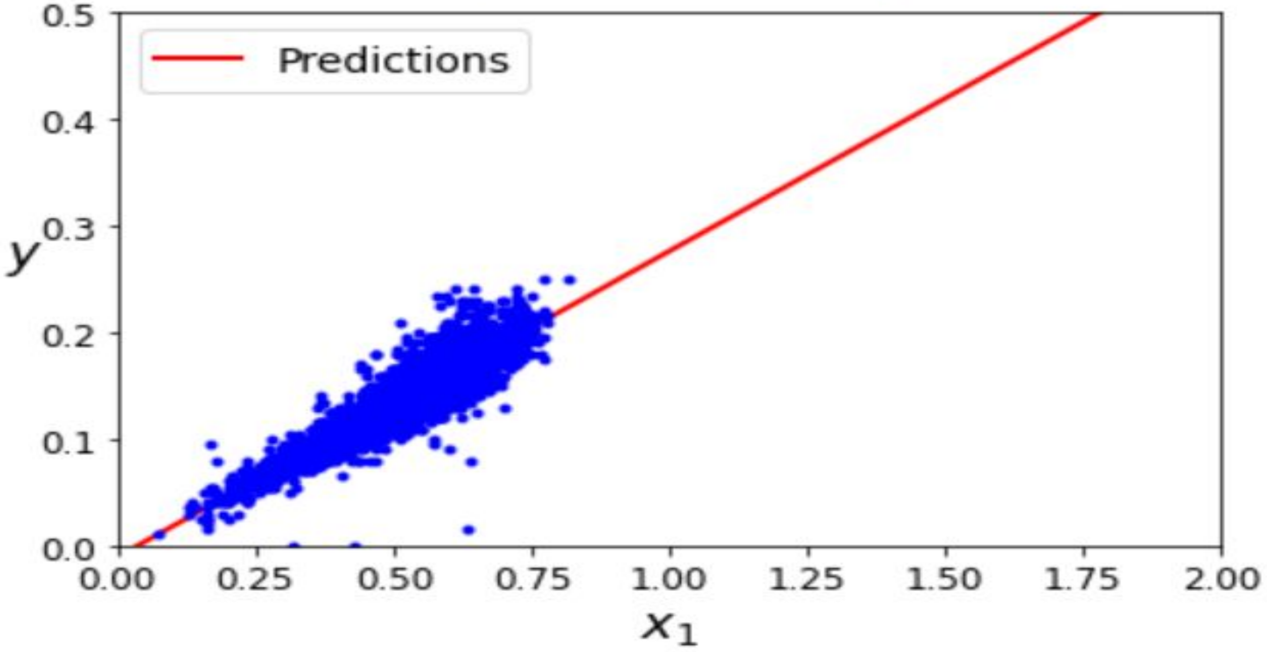
Test

1s

```
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()
```

Test

Saving figure linear_model_predictions_plot



Enhancement Ideas

- Regularization techniques, such as L1 or L2 regularization, can be applied to the normal equation solution to prevent 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 for handling large datasets that fit into the computer's memory.

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

References

❖ *Kwiatkowski, R. (2020, December 04). Performing linear regression using the normal equation. Retrieved February 4, 2023, from https://towardsdatascience.com/performing-linear-regression-using-the-normal-equation-6372ed3c57*

❖ *Ageron. (2021, October 17). Handson-ML2/04_training_linear_models.ipynb at master · Ageron/Handson-ML2. Retrieved February 4, 2023, from https://github.com/ageron/handson-ml2/blob/master/04_training_linear_models.ipynb*