Q28

1. Follow the procedure mentioned in Chapter 4 – Training Linear Models to make it work on Colab.
2. Save the abalone_train.cvs to a local drive
    - Note: the abalone_train.cvs has this format
        -
        - names=["Length", "Diameter", "Height", "Whole weight", "Shucked weight",
        - "Viscera weight", "Shell weight", "Age"])
3. Change the process mentioned in Step 1 by reading CVS test data from a local drive : abalone_train.cvs
    - Process
        a. You can modify the code in Linear regression using the Normal Equation. Instead of reading random data

You need to read data from a local drive and transfornm the data to fit the Python code.

```
import numpy as np
import pandas as pd

# X = 2 * np.random.rand(100, 1)
# y = 4 + 3 * X + np.random.randn(100, 1)
from google.colab import files
uploaded  = files.upload()

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"])
# X1 is
#    0        0.435
#    1        0.585
#    2        0.655
#    .....
X1 = abalone["Length"]

# X2 is
#    array([0.435, 0.585, ...., 0.45])
X2 = np.array(X1)

# X is
#    array([[0.435],
#           [0.585],
#           [0.655],
#           ...,
```

```
#            [0.53 ],
#            [0.395],
#            [0.45 ]])
X = X2.reshape(-1, 1)

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

Answer:

Go to colab

Setup

First, let's import a few common modules, ensure MatplotLib plots figures inline and prepare a
Python 3.5 or later is installed (although Python 2.x may work, it is deprecated so we strongly
Scikit-Learn ≥0.20.

```
# Python ≥3.5 is required
import sys
assert sys.version_info >= (3, 5)

# Scikit-Learn ≥0.20 is required
import sklearn
assert sklearn.__version__ >= "0.20"

# Common imports
import numpy as np
import os

# to make this notebook's output stable across runs
np.random.seed(42)

# To plot pretty figures
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)

# Where to save the figures
PROJECT_ROOT_DIR = "."
```

Change code

Linear regression using the Normal Equation

```python
import numpy as np
import pandas as pd

X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)
from google.colab import files
uploaded  = files.upload()

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"])
# X1 is
#    0       0.435
#    1       0.585
#    2       0.655
#    .....
X1 = abalone["Length"]

# X2 is
#    array([0.435, 0.585, ...., 0.45])
X2 = np.array(X1)

# X is
#    array([[0.435],
#           [0.585],
#           [0.655],
#           ...,
```

Upload file

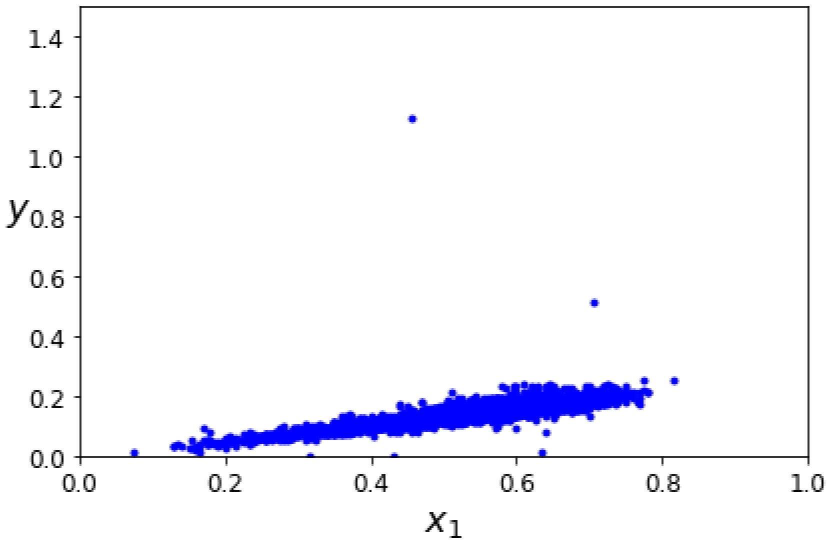
```
# X2 is
#    array([0.435, 0.585, ...., 0.45])
X2 = np.array(X1)

# X is
#    array([[0.435],
#           [0.585],
#           [0.655],
#           ...,
#           [0.53 ],
#           [0.395],
#           [0.45 ]])
X = X2.reshape(-1, 1)

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

```
Choose Files abalone_train.csv
• abalone_train.csv(application/vnd.ms-excel) - 149229 bytes, last modified: 6/2/2021 - 100% done
Saving abalone_train.csv to abalone_train.csv
```

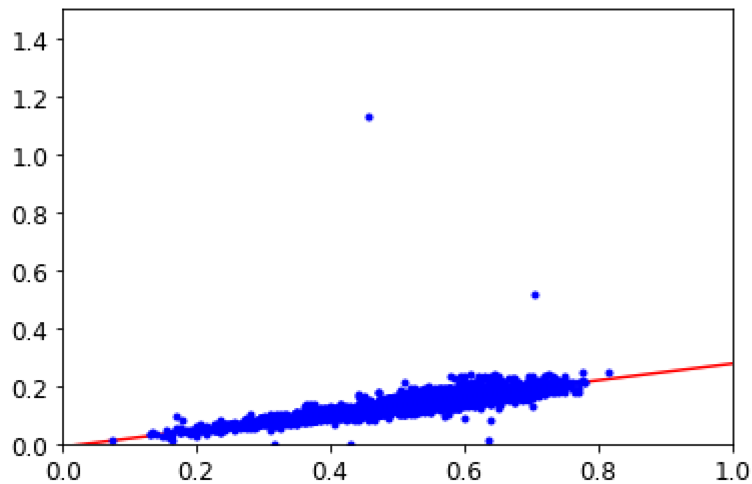
The result

```
plt.plot(X, y, "b.")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.axis([0, 1, 0, 1.5])
save_fig("generated_data_plot")
plt.show()
```

```
Saving figure generated_data_plot
```

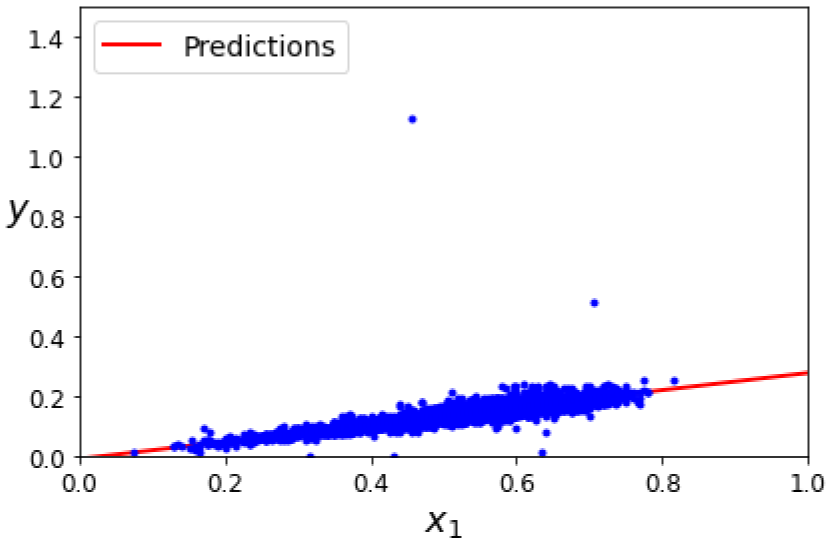


```python
plt.plot(X_new, y_predict, "r-")
plt.plot(X, y, "b.")
plt.axis([0, 1, 0, 1.5])
plt.show()
```



```
[14] 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, 1, 0, 1.5])
     save_fig("linear_model_predictions_plot")
     plt.show()
```

```
Saving figure linear_model_predictions_plot
```



[15]
```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(X, y)
lin_reg.intercept_, lin_reg.coef_
```

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

[16]
```
lin_reg.predict(X_new)
```

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

The `LinearRegression` class is based on the `scipy.linalg.lstsq()` function (the name stands for "least squares"), which you could call directly:

```
theta_best_svd, residuals, rank, s = np.linalg.lstsq(X_b, y, rcond=1e-6)
theta_best_svd
```

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

This function computes $\mathbf{X}^{+}\mathbf{y}$, where $\mathbf{X}^{+}$ is the *pseudoinverse* of $\mathbf{X}$ (specifically the Moore-Penrose inverse). You can use `np.linalg.pinv()` to compute the pseudoinverse directly:

[18]
```
np.linalg.pinv(X_b).dot(y)
```

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