Linear Regression on a Simulated Dataset

Table of contents

1 About

This PDF shows applying a linear regression model on a simulated datset of 3 features and 1 target variable. The purpose is to demonstrate the conceptual understanding of a linear algebra interpretation of the linear regression model.

2 Data

2.1 simulation and exporting

The data is simulated using the following code:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


feature_1 = np.random.normal(5, 3, 1000)
feature_2 = np.random.normal(10, 5, 1000)
feature_3 = np.random.normal(-7, 1.5, 1000)
```

Note

Food for thought: what if the features are not normally distributed?

```python
beta_0, beta_1, beta_2, beta_3 = 10, 5, 3, 1
```

```python
feature_1 = feature_1 * beta_1
feature_2 = feature_2 * beta_2
feature_3 = feature_3 * beta_3

y = beta_0 + feature_1 + feature_2 + feature_3

## add some error to y
y = y + np.random.normal(2, 1, 1000)

df = pd.DataFrame({'feature_1': feature_1, 'feature_2': feature_2, 'feature_3': feature_3,
```

```python
## taking a peak at the data
df.sample(5)
```

	feature_1	feature_2	feature_3	y
209	5.299494	16.323631	-6.251507	24.323130
854	32.426386	51.585037	-8.873245	87.797870
762	32.241610	36.940929	-6.120289	75.007230
516	23.304671	50.176325	-6.144909	80.273822
707	14.759471	35.669615	-8.237580	53.591483

```python
## export the dataset
df.to_csv('dataset.csv', index=False)
```

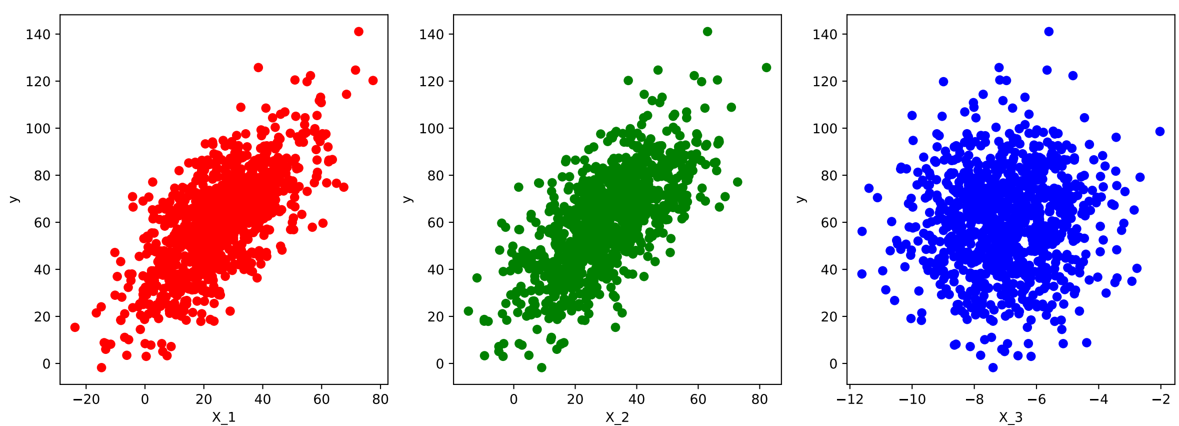
2.2 importing the dataset

```
## read the dataset
data = pd.read_csv('dataset.csv')
```

```
## rename columns as X_1, X_2, X_3
data.columns = ['X_1', 'X_2', 'X_3', 'y']
```

2.3 Checking the assumption of linearity

```
## create a scatter plot of each feature vs the target variable in subplots
fig, ax = plt.subplots(1, 3, figsize=(15, 5))

ax[0].scatter(data['X_1'], data['y'], color='red')
ax[0].set_xlabel('X_1')
ax[0].set_ylabel('y')

ax[1].scatter(data['X_2'], data['y'], color='green')
ax[1].set_xlabel('X_2')
ax[1].set_ylabel('y')

ax[2].scatter(data['X_3'], data['y'], color='blue')
ax[2].set_xlabel('X_3')
ax[2].set_ylabel('y')

plt.show()
```



2.4 splitting the data

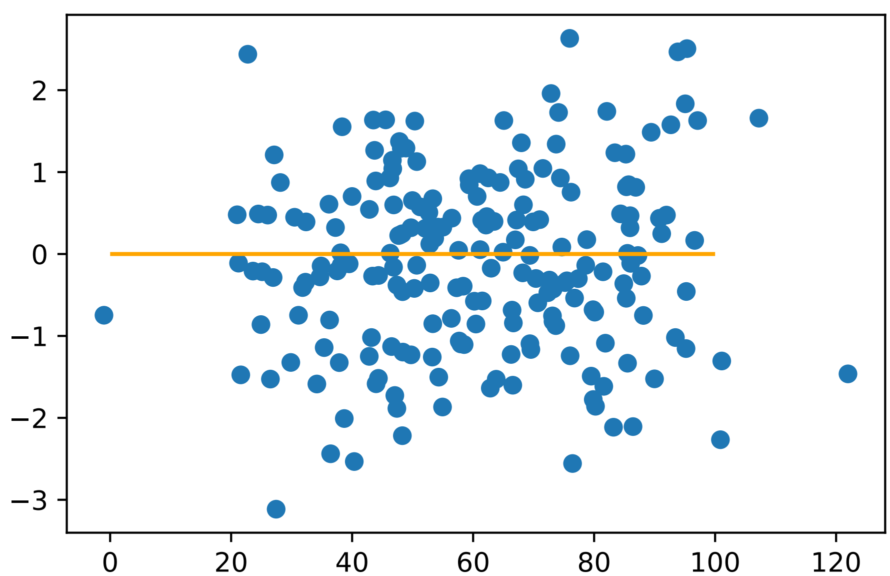
```
## split the data into training and testing sets
from sklearn.model_selection import train_test_split

X = data[['X_1', 'X_2', 'X_3']]
y = data['y']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

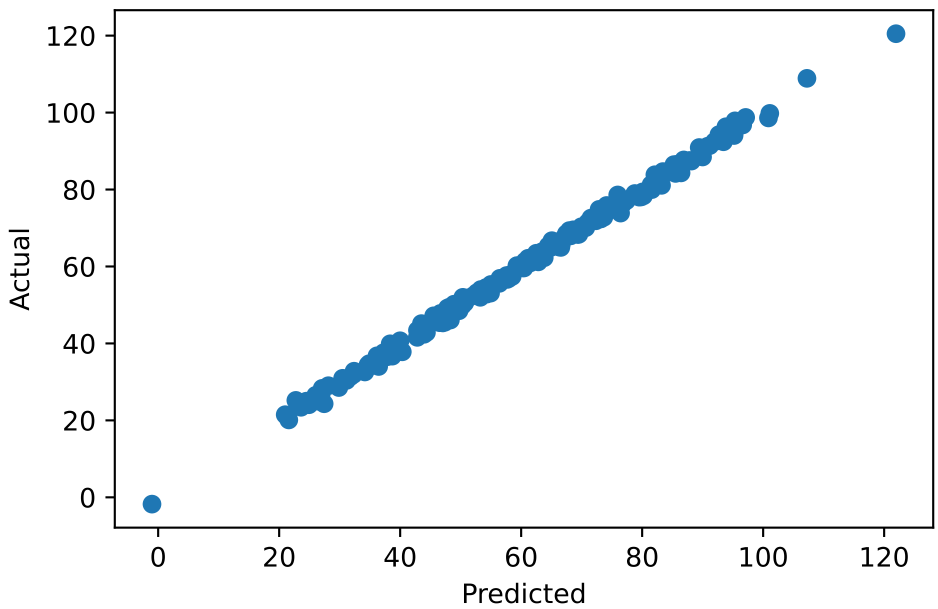
2.5 applying the Linear Regression model and evaluating the model

```
## fit the model
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)

## print the coefficients and intercept
print(model.coef_)
print(model.intercept_)

## predict on the test data
y_pred = model.predict(X_test)

## calculate the mean squared error
from sklearn.metrics import mean_squared_error

print(mean_squared_error(y_test, y_pred))

## calculate the R-squared
from sklearn.metrics import r2_score

print(r2_score(y_test, y_pred))

## plot the residuals

plt.scatter(y_pred, y_test - y_pred)

plt.hlines(y=0, xmin=0, xmax=100, color='orange')
```

```
plt.show()

## plot predictions vs actual

plt.scatter(y_pred, y_test)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()
```

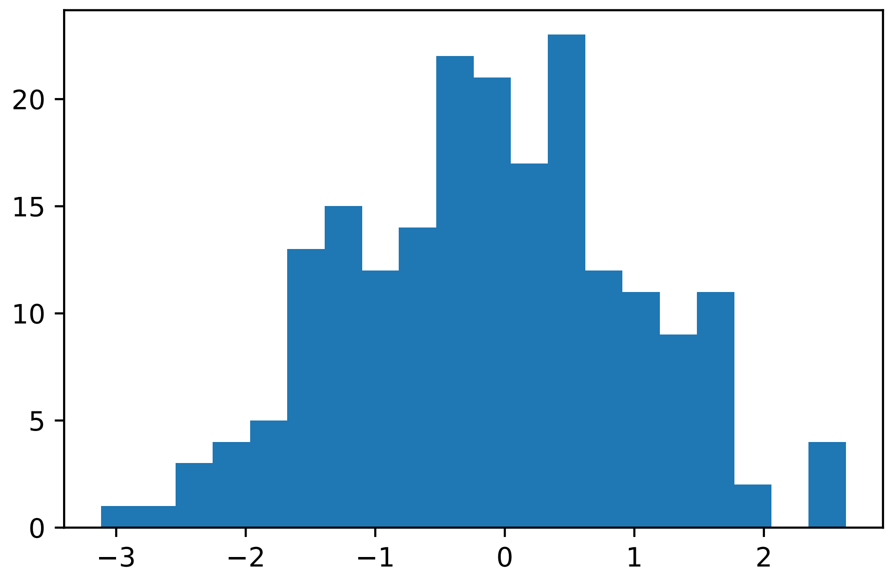
```
[1.00061402 0.99723559 1.01272485]
12.187570233683047
1.2210607773999902
0.9972973878411193
```





2.6 checking the distribution of the residuals visually

```
## plot the residuals

plt.hist(y_test - y_pred, bins=20)

plt.show()
```



2.7 checking the distribution of the residuals numerically

```
## check the distribution of the residuals numerically
## call wolfram client from python

from wolframclient.evaluation import WolframLanguageSession
from wolframclient.language import wl, wlexpr
session = WolframLanguageSession("J:\installed\WolframKernel.exe")

##def find_distribution(data):
##    return session.evaluate(wl.FindDistribution(data))

residuals = np.array(y_test - y_pred)

session.evaluate(wl.FindDistribution(residuals))
```

```
NormalDistribution[-0.10882693978803204, 1.131127132901281]
```

3 Further actions:

1. The data can be from distributions other than the normal distribution.

2. I can omit the noise data and see what coefficients are estimated.