Machine Learning

Sample Project (POGO Showcase)

Curtis Larsen

Utah Tech University—Computing

Spring 2025

Objectives

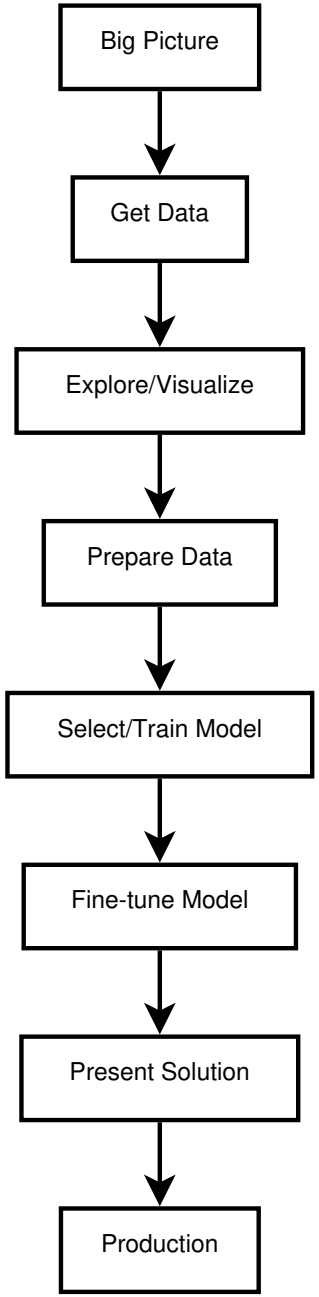
Objectives:

- Understand Project Process
- Apply Process to Sample Problem

Project Outline

Outline:

- Look at the big picture.
- Get the data.
- Explore and visualize to gain insights.
- Prepare the data for machine learning algorithms.
- Select a model and train it.
- Fine-tune your model.
- Present your solution.
- Launch, monitor, and maintain your system.

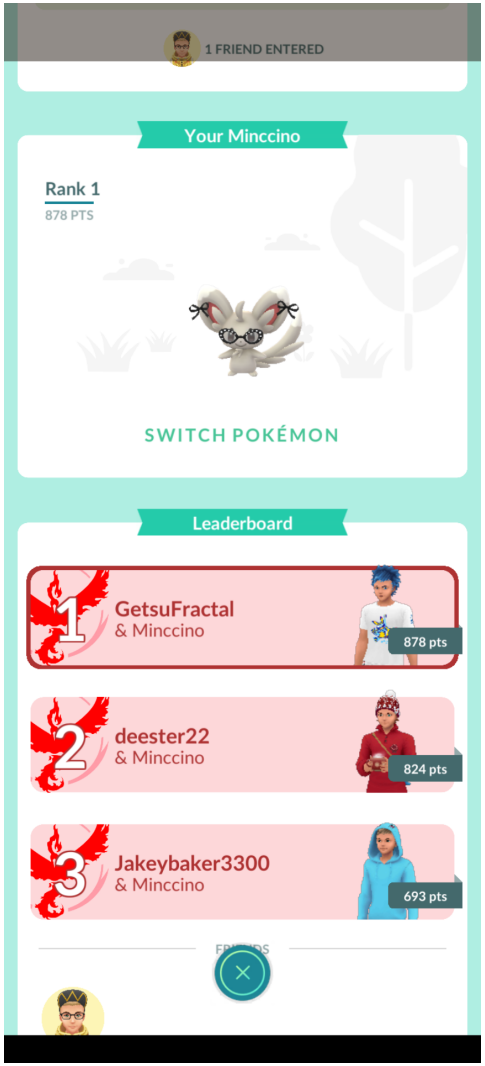


Big Picture

POGO Showcases

Principles:

- Showcase large pokemon.
- Constrained to species.
- In-game rewards.
- Formula for score unknown.
- Saving pokemon for future use.



Pokemon Stats

Principles:

- Various stats for each pokemon.
- Showcase score is not shown.
- Formula is not disclosed.
- Weight and Height are suspected inputs to formula.
- CP and other stats may also be inputs?

Scores

Princples:

- After entering, scores can be seen.
- Only for pokemon currently in storage.

Purpose

Given a pokemon's stats, predict its showcase score, for any possible showcase it may be eligible for in the future.

Get Data

Data

Data:

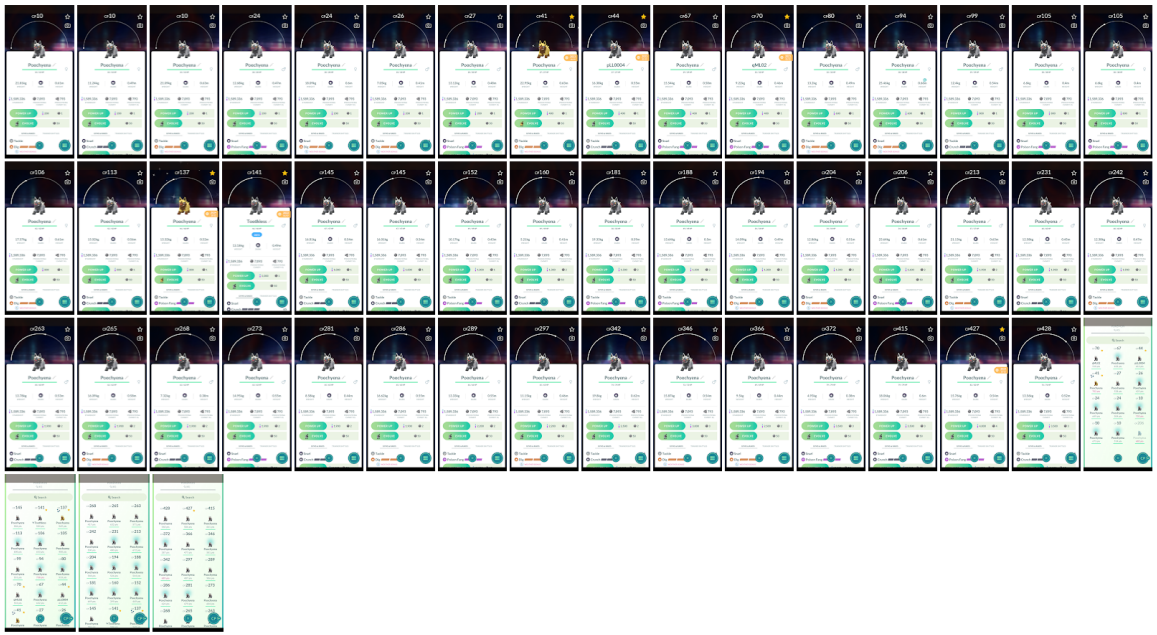
- Showcase scores.
- Extract for as many pokemon as possible.

Data

Data:

- Pokemon stats.
- Extract for as many pokemon as possible.
- Match with score data.

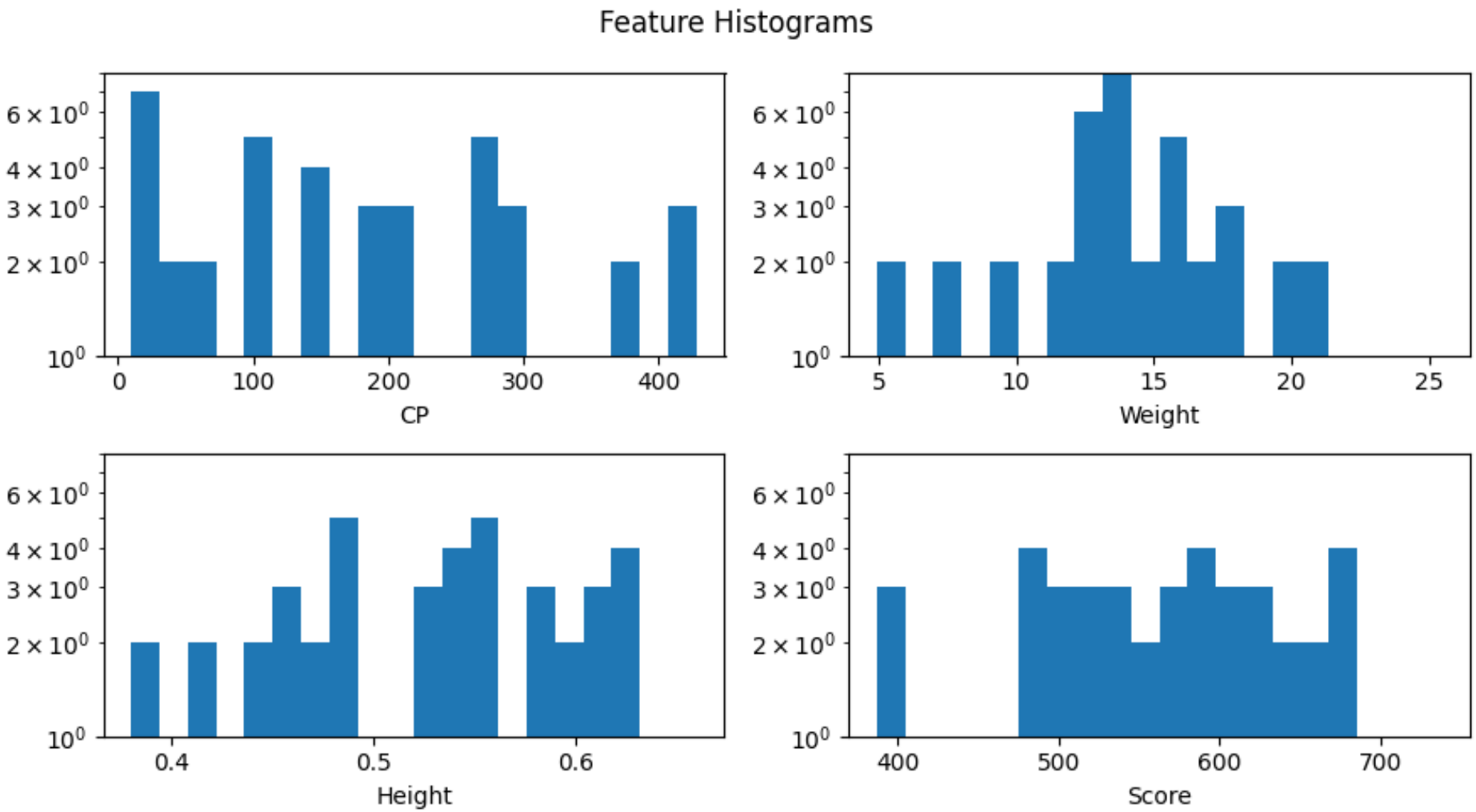
Data



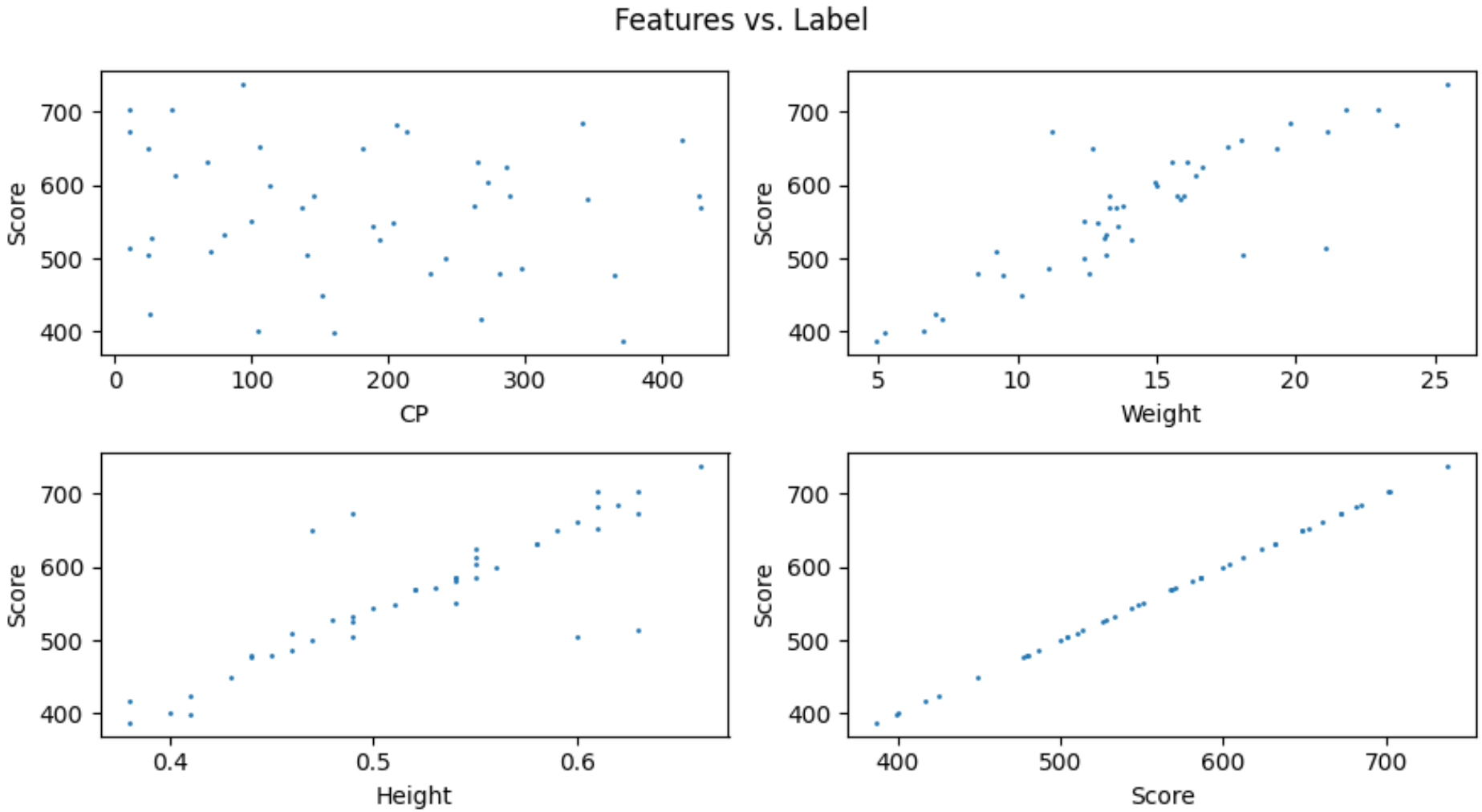
	A	B	C	D	E
1	Species	CP	Weight	Height	Score
2	Poochyena	10	21.81	0.61	703
3	Poochyena	10	11.24	0.49	673
4	Poochyena	10	21.09	0.63	514
5	Poochyena	24	12.68	0.47	649
6	Poochyena	24	18.09	0.6	504
7	Poochyena	26	7.05	0.41	425
8	Poochyena	27	13.12	0.48	528
9	Poochyena	41	22.95	0.63	702
10	Poochyena	44	16.38	0.55	612
11	Poochyena	67	15.54	0.58	632
12	Poochyena	70	9.22	0.46	510
13	Poochyena	80	13.2	0.49	533
14	Poochyena	94	25.46	0.66	738
15	Poochyena	99	12.4	0.54	551
16	Poochyena	105	6.6	0.4	400

Explore/Visualize

Histograms



Scatter Plots

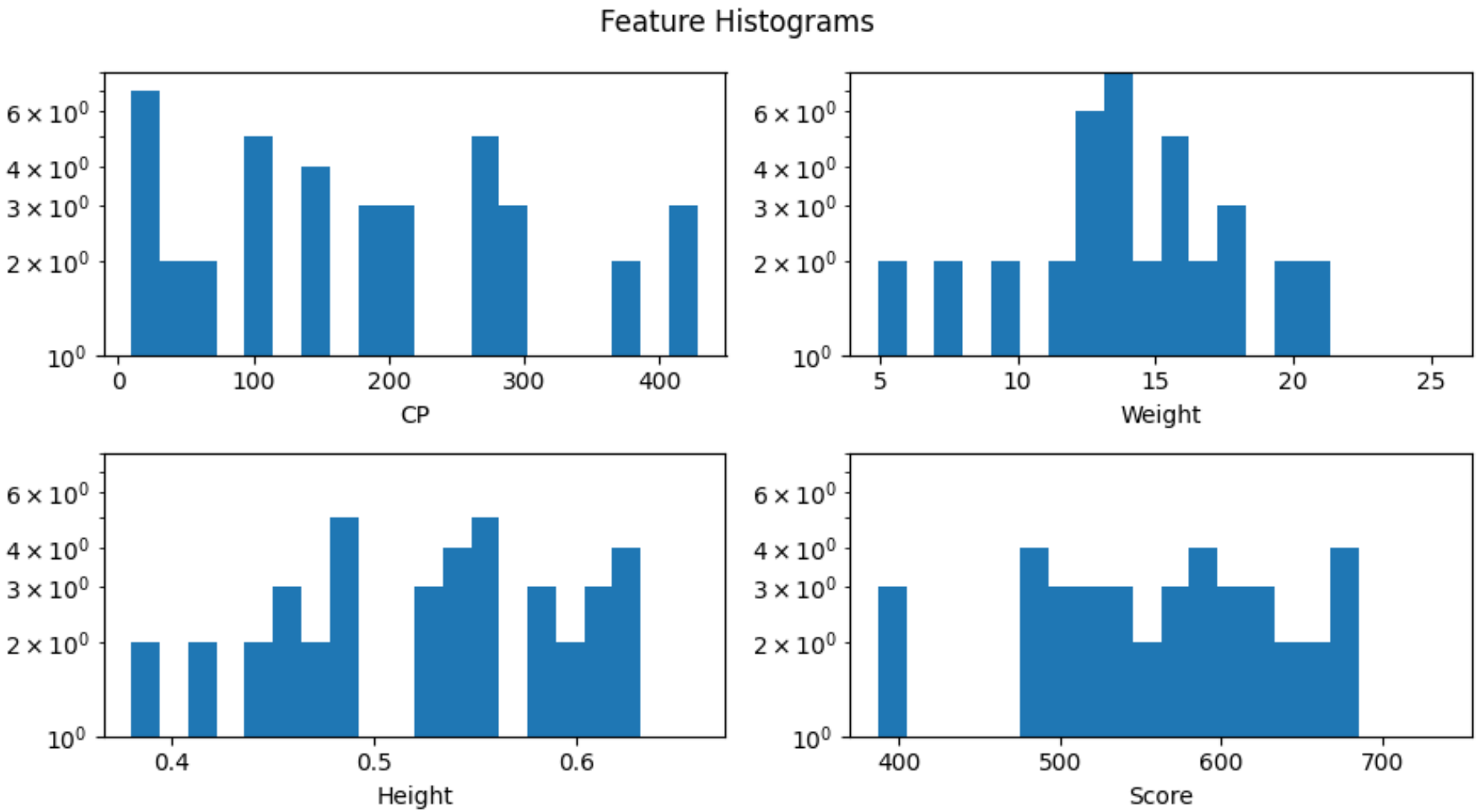


Prepare

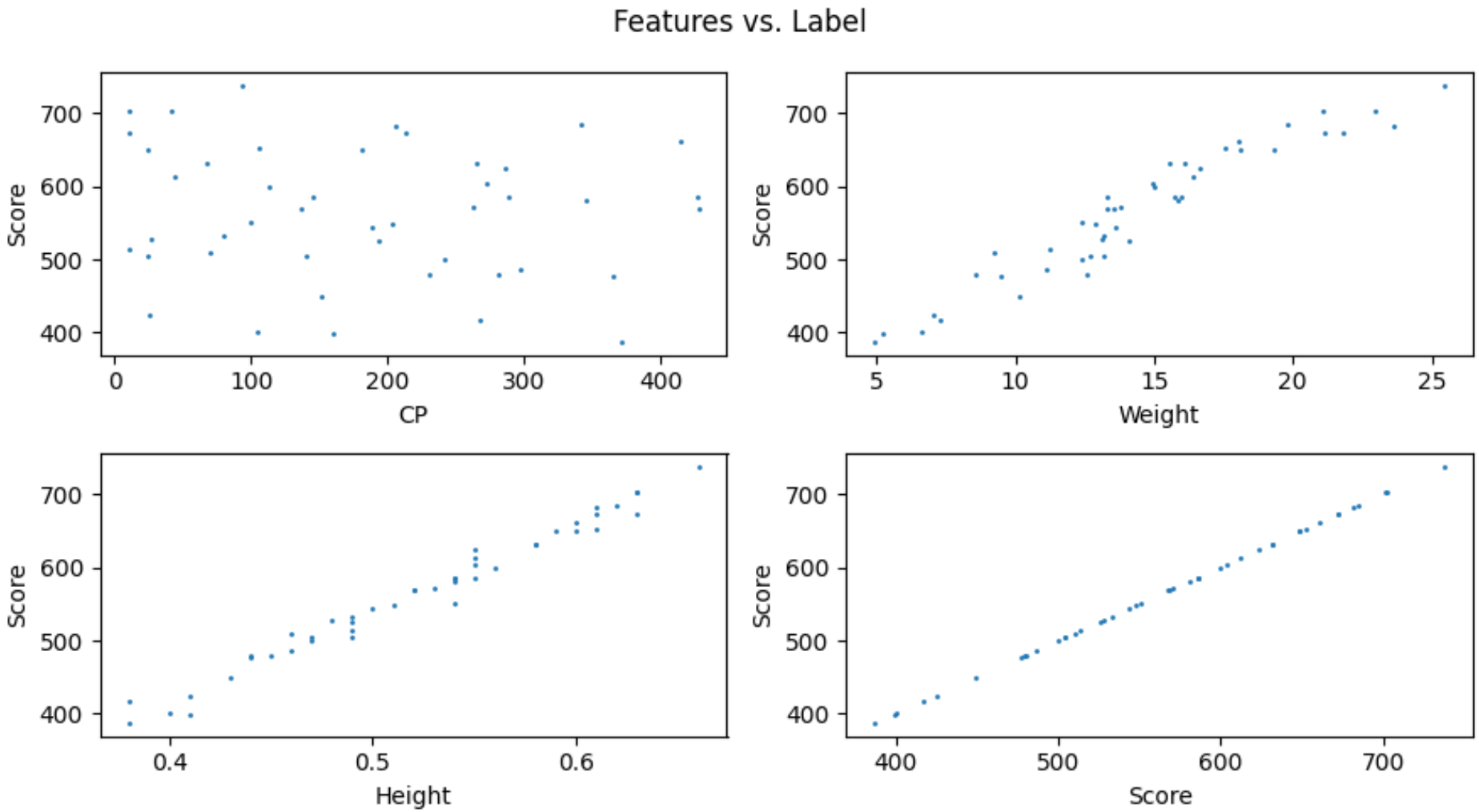
Steps

- Fix entry errors.
- Remove outliers/errors.
- Split into training/testing data.

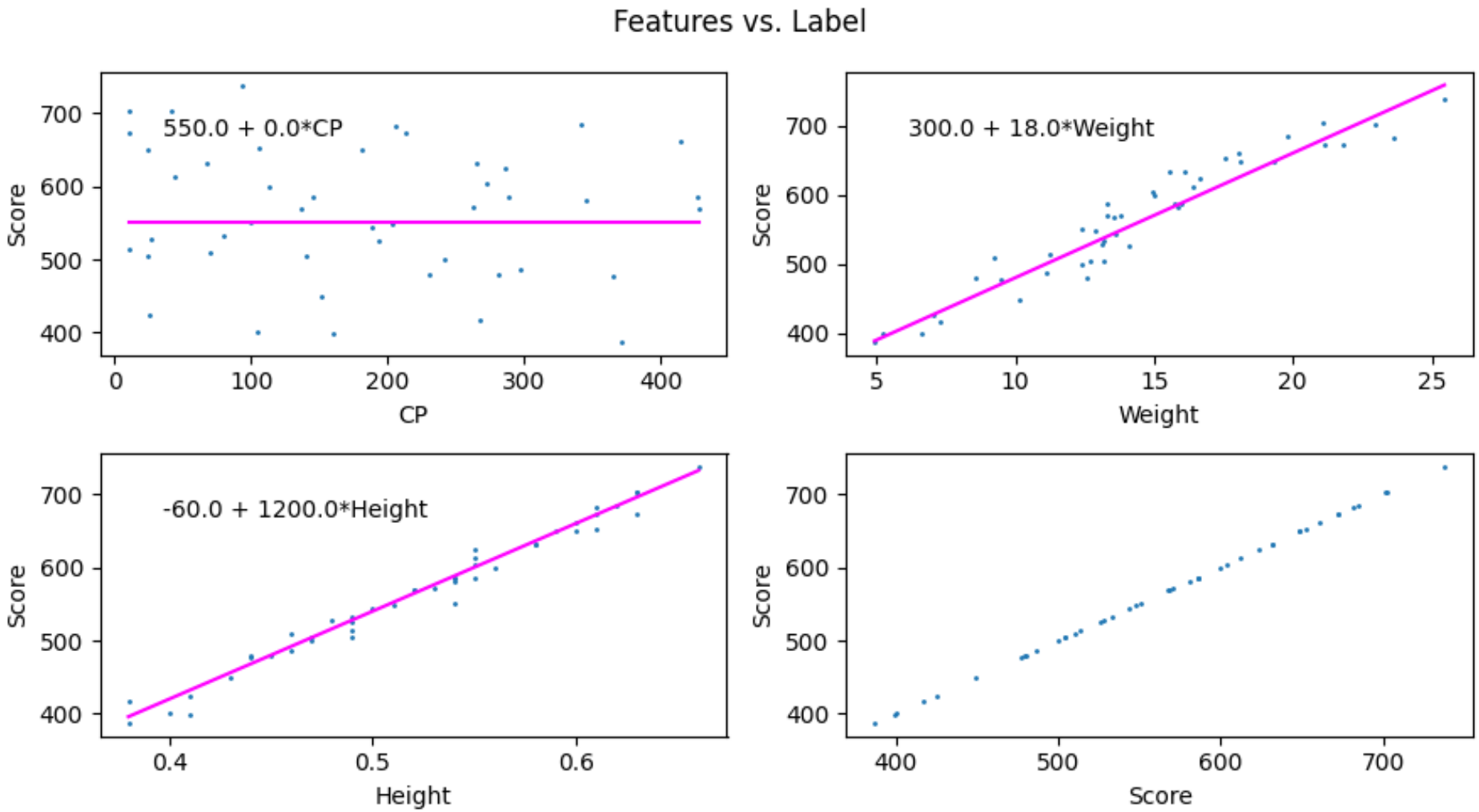
Histograms



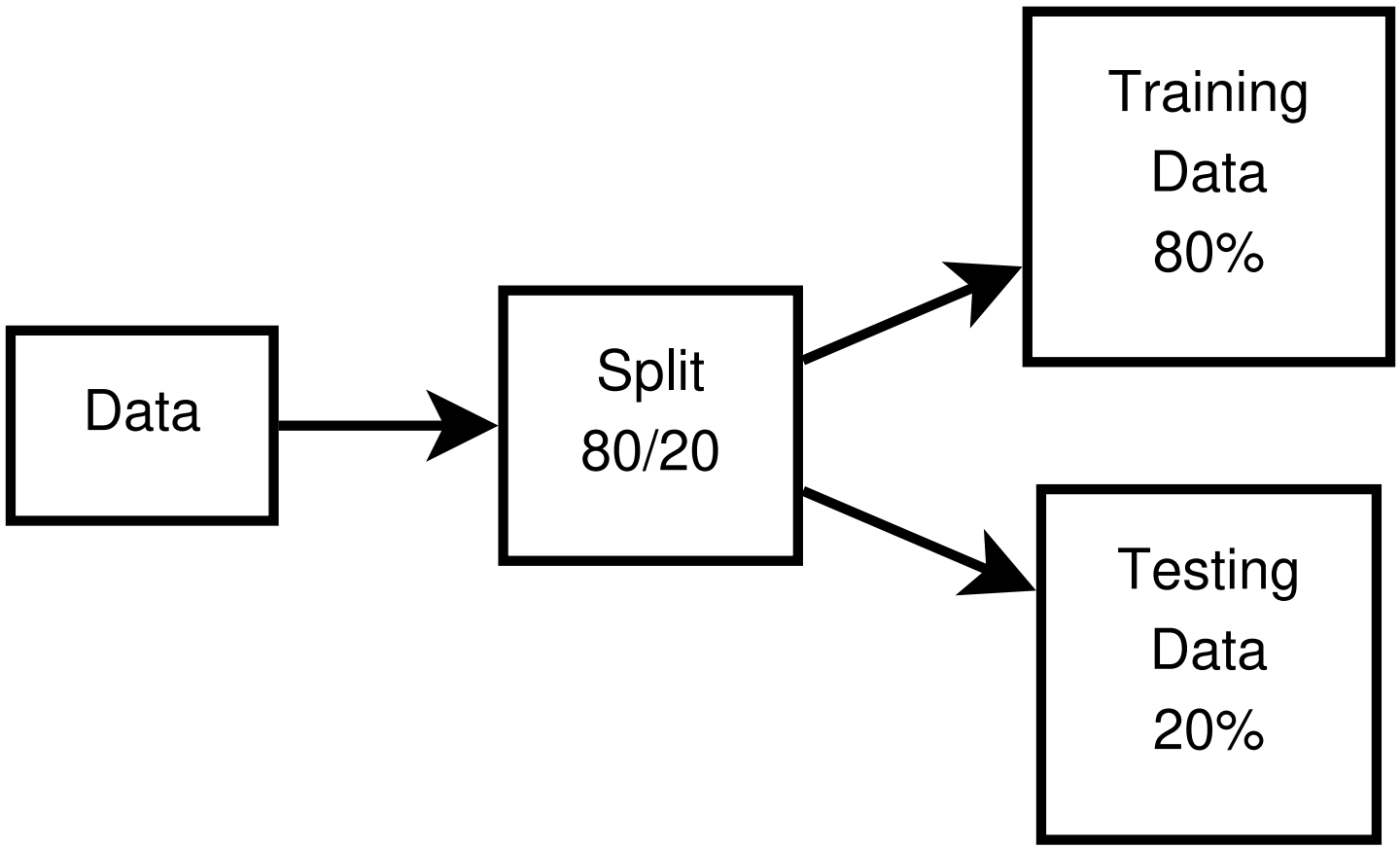
Scatter Plots



Visual Guess Model



Split Data



Split Data

```
import pandas as pd
import sklearn

filename = "showcase-prepared.csv"
train_filename = "showcase-prepared-train.csv"
test_filename = "showcase-prepared-test.csv"
data = pd.read_csv(filename)
seed = 42
ratio = 0.2
data_train, data_test = \
    sklearn.model_selection.train_test_split(
        data, test_size=ratio, random_state=seed)
data_train.to_csv(train_filename)
data_test.to_csv(test_filename)
```

Select/Train Model

Steps

- Choose linear regression model, based on scatter plots
- Choose not to include feature `CP` based on scatter plots
- Build program to load data, train model, and save model.

$$y = f_\theta(\vec{x}) = \sum_{i=0}^{n} \theta_i x_i = \theta_0 + \theta_1 w + \theta_2 h$$

$$x_0 = 1$$

Load Data

```
import pandas as pd

feature_names = ["Weight", "Height"]
label_name = "Score"
train_filename = "showcase-prepared-train.csv"
data = pd.read_csv(train_filename, index_col=0)
X_train = data[feature_names]
y_train = data[label_name]

model_filename = "showcase-linear-regressor.joblib"
```

Fit Model

```
import sklearn
import joblib

# read data, define fields, etc.
from showcase_common import *

# peek at data
print(data.head(5))

# do the fit/training
regressor = sklearn.linear_model.SGDRegressor(verbose=1)
regressor.fit(X_train, y_train)

# save the trained model
joblib.dump(regressor, model_filename)
```

View Training Score

```
import sklearn
import joblib

# read data, define fields, etc.
from showcase_common import *

# load model
regressor = joblib.load(model_filename)

# ask model to score the data
score_train = regressor.score(X_train, y_train)
print("R^2: {}".format(score_train))
```

View Training Score

```
# show mean square error and mean absolute error
y_predicted = regressor.predict(X_train)
loss_train = \
  sklearn.metrics.mean_squared_error(y_train,
                                     y_predicted)
print("MSE: {}".format(loss_train))
loss_train = \
  sklearn.metrics.mean_absolute_error(y_train,
                                      y_predicted)
print("MAE: {}".format(loss_train))
```

Metrics

```
R^2: 0.06910893098422
MSE: 6937.766651221209
MAE: 62.22596751638615
```

Showing Model

```
import joblib

# read data, define fields, etc.
from showcase_common import *

regressor = joblib.load(model_filename)

def show_model(regressor):
    print("Model Information:")
    print("coef_: {}".format(regressor.coef_))
    print("intercept_: {}".format(regressor.intercept_))
    print("n_iter_: {}".format(regressor.n_iter_))
    print("n_features_in_: {}".format(regressor.n_features_in
    return
```

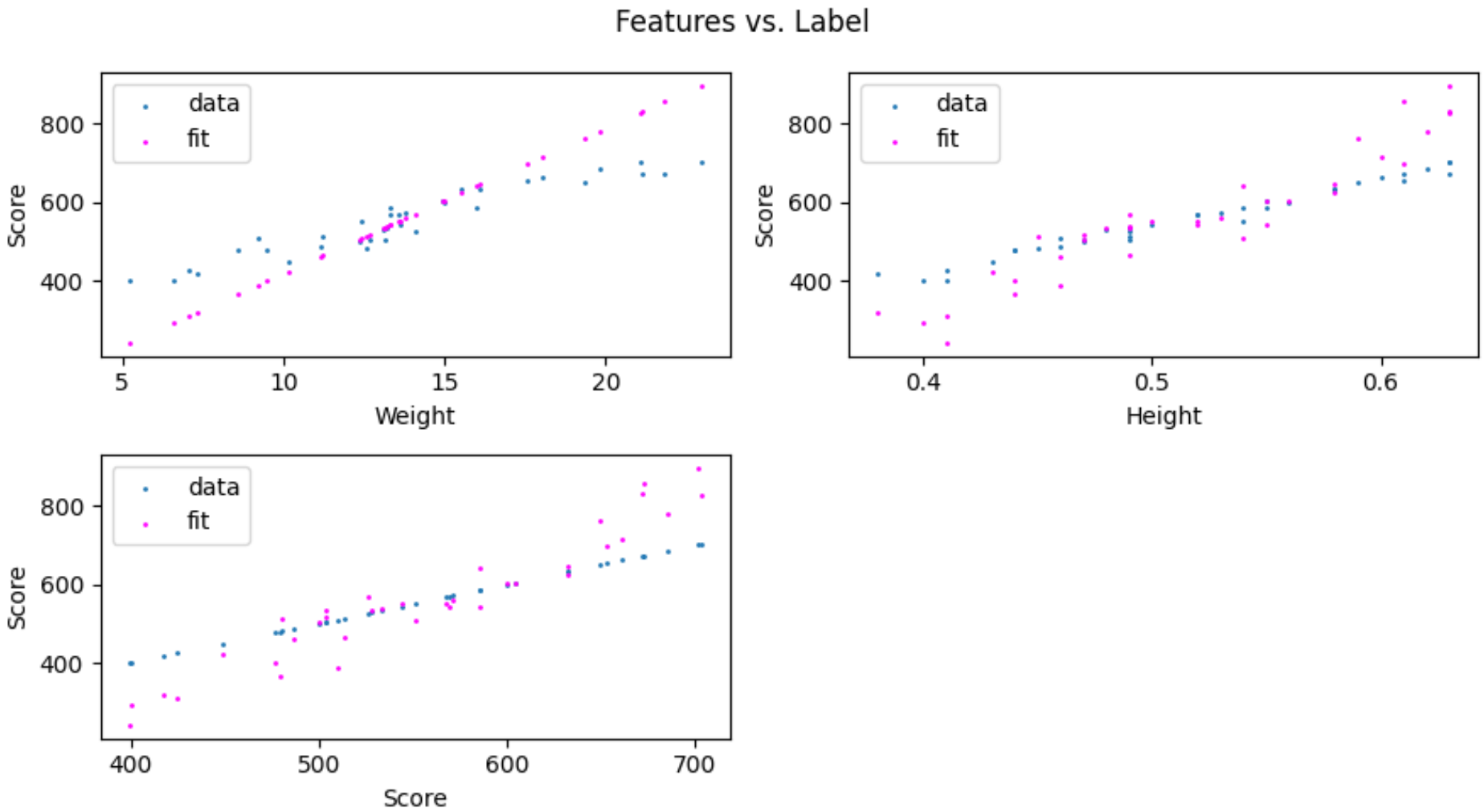
Showing Function

```
def show_function(regressor):
    offset = regressor.intercept_[0]
    s = "{:6.3f}".format(offset)

    # term for each feature
    for i in range(0, len(regressor.coef_)):
        if len(feature_names[i]) > 0:
            t = "({:6.3f}*{})".format(regressor.coef_[i], fea
        if len(s) > 0:
            s += " + "
        s += t

    print("Function:")
    print("{}".format(s))
    return
```

Trained Model

```
Model Information:
coef_: [36.83331156 14.31547234]
intercept_: [42.80714754]
n_iter_: 14
n_features_in_: 2

Function:
42.807 + (36.833*Weight) + (14.315*Height)
```

Visualize Fit Model



Fine-tune Model

Steps

- Linear regression algorithm works better with normalized and scaled features.
- `CP` *may* have a minor effect. Add it back in.

Data Column Scaling

Mean $\mu = \frac{\sum_{j=1}^{m} x_j}{m}$

Variance $v = \frac{\sum_{j=1}^{m} (x_j - \mu)^2}{m}$

Standard Deviation $\sigma = \sqrt{v}$

Scaled Data $z_j = (x_j - \mu)/\sigma$

- New mean is 0.
- New standard deviation is 1.

Metrics

```
R^2: 0.9828144868112312
MSE: 128.0805931581468
MAE: 8.589027939544692
```

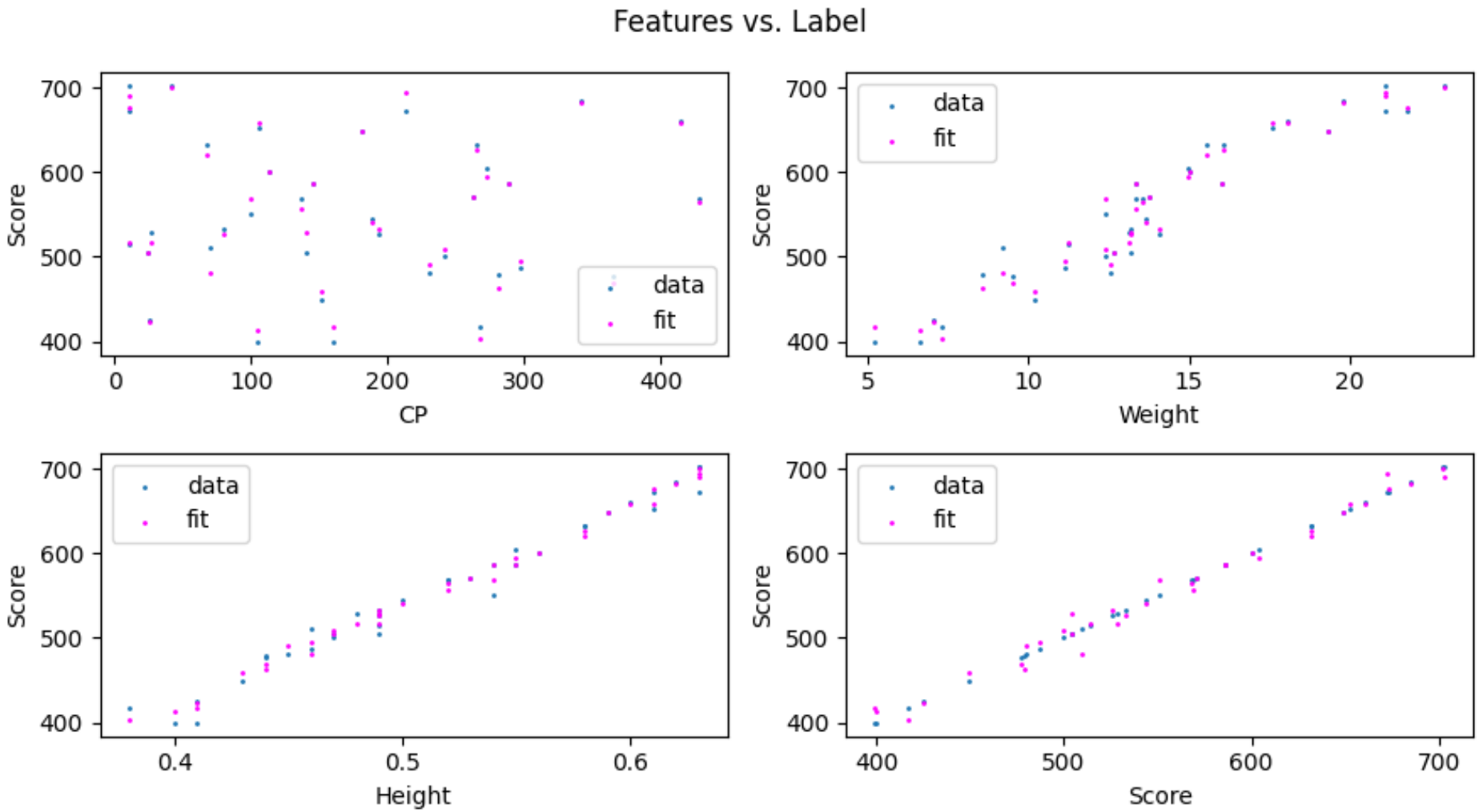
Trained Scaler

```
Scaler Information:
scale_: [1.16734065e+02 4.36162838e+00 7.19369365e-02]
mean_: [173.86111111  13.79583333   0.51527778]
var_: [1.36268418e+04 1.90238021e+01 5.17492284e-03]
feature_names_in_: ['CP' 'Weight' 'Height']
```

Trained Model

```
Model Information:
coef_: [ 2.58525445 20.09967675 66.33773433]
intercept_: [554.80347683]
n_iter_: 1311
n_features_in_: 3

Function with scaler corrections:
12.207 + ( 0.022*CP) + ( 4.608*Weight) + (922.165*Height)
```

Visualize Fit Model



Present Solution

Steps

- Score on testing data

Training vs Testing Metrics

Training Metrics:

```
R^2: 0.9828144868112312
MSE: 128.0805931581468
MAE: 8.589027939544692
```

Testing Metrics:

```
R^2: 0.9876210226656025
MSE: 106.61539414786523
MAE: 7.913890499120701
```

Production

Steps

-
-
-